

PERSONAL COMPUTER CLUB

MENSILE DI PERSONAL E HOME COMPUTER

30

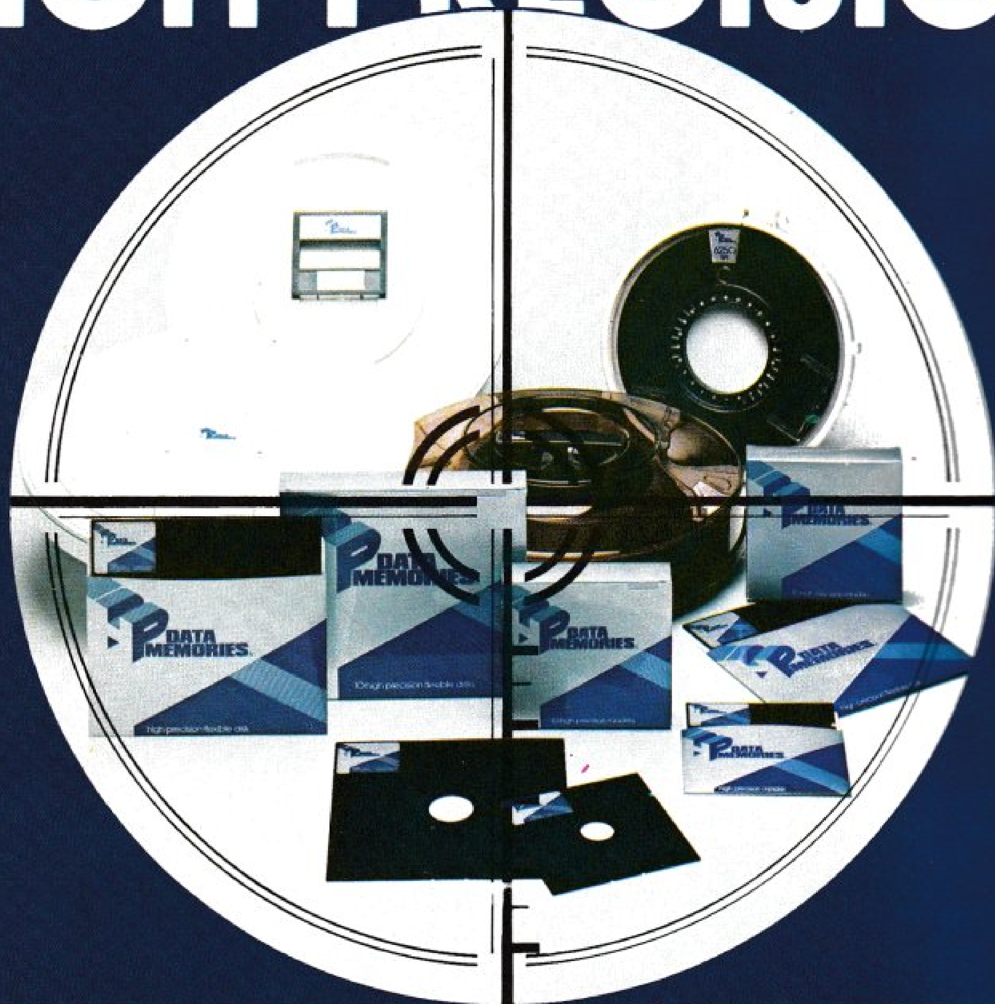
Programmi

per il tuo

Commodore

Supplemento al N. 19 di Personal Computer
Lit. 3.500

MEE OBIETTIVO HIGH PRECISION





High precision Data Memories
é tecnologia avanzata di costruzione.
È il supporto magnetico testato ai limiti della resistenza con garanzia di assoluta affidabilità.
È avanguardia tecnologica per assicurare la massima protezione dei dati, anche, nelle situazioni più critiche.

HIGH PRECISION A COLPO SICURO!



MEE - Memorie per Elaboratori Elettronici S.p.A.
Forniture per Centri Elaborazione Dat
Sede Amm.va: 20144 Milano - Via Bon 29
Tel. 4988541 (4 linee r.a.) - Telex 324426 MEE-
Filiali e Agenzie: Milano - Bergamo - Torino
Biella - Padova - Parma - Bologna - Firenze - Ancona
Roma - Napoli - Catania - Oristano - Bari - Genova
Bolzano - Mestre



PERSONAL COMPUTER CLUB
MENSILE DI
PERSONAL COMPUTING
Anno II - N. 19 - dicembre 1984
Sped. Abb. Post. Gr. III (70%)
Reg. Trib. Milano: n. 262 del 4/5/1983
Una copia Lit. 3.000
arretrati il doppio da richiedere alla
Casa Editrice con pagamento anticipato
SOCIETÀ EDITRICE:
ALFA LINEA
Direzione, redazione
Piazza Cavour, 2 - 20121 Milano
Telefono: (02) 782661/2/3
DIRETTORE RESPONSABILE
Marco Bindi
Redazione
Paolo Tampieri
Marinella Zetti
Collaboratori
Giacomo Buico
Leonardo Felician
Matteo Ferrari
Pier Luigi Maggi
Angelo Magistri
Roberto Marconcini
Roberto Mariani
Gianroberto Negri
Walter Pistarini
Edoardo Piva
Ernesto Sagramoso
Giordano Serafin
DIREZIONE COMMERCIALE
Via Anfiteatro, 15 - 20121 Milano
Tel. (02) 802388/876622/866220
Direttore Commerciale
Gianluca Rivoli
Direzione Pubblicità
Via Anfiteatro, 15 - 20121 Milano
Tel. (02) 802388/876622/866220
Daniela Morandi **(Responsabile)**
Margherita Del Monaco
Ketty Cusin
Emanuela Manni
Tiziana Belotti **(segreteria)**
Conc. escl. Pubblicità
per Roma e Lazio
ALFA MEDIA
Via L. Signorelli, 11
00196 Roma - Tel.: (06) 3963942
Marcella Casagni
M. Enrica Castelletti
Diffusione-Abbonamenti
Via Anfiteatro, 15 - 20121 Milano
Tel. (02) 8059425
Vanda Zaglio
AMMINISTRAZIONE
Via Anfiteatro, 15 - 20121 Milano
Tel.: (02) 8059425
Milena Collica
Fotocomposizione: Proget (Milano)
Stampa: Rotolito Lombarda S.p.A. (Milano)
Distribuzione per l'Italia: Messaggerie Periodici
via G. Carcano 32 - Milano
Sole agent for distribution abroad: A.I.E. - Agenzia Italiana di Esportazione S.p.A. - Via Gadames, 89 20151 Milano - Telefono 30.12.200 - Telex 315367.

30 programmi per il tuo Commodore

Supplemento al N. 19 di Personal Computer - Lit. 3.500

SOMMARIO

Scrivetevi le vostre avventure!

di Walter Pistarini

«Adventure» è il nome di un gioco (o di un tipo di gioco) dove vi trovate ad esplorare una fitta rete di stanze, grotte, labirinti, colline o altro, in cerca di tesori senza prezzo. Qualche nome: Adventure, Dungeon & Dragons, Hobbit, Mad Martha.

Il primo gioco di questo tipo comparve negli anni '70 sui grossi calcolatori. Le sue origini non sono pienamente accertate, ma i più informati tendono a darne la paternità a Willie Crowther, dell'Università di Stanford, che scrisse il programma in Fortran. In seguito questa versione originale è stata riveduta e modificata da Don Woods. Fino a quel momento, e per lungo tempo, esplorare i meandri di questo gioco richiedeva l'accesso ai grandi calcolatori, visto che occorreva una disponibilità di almeno 200.000 bytes di memoria. Arrivò un bel giorno, eravamo nel 1978, in cui un tipo destinato a diventare famoso si mise in testa di riscrivere il gioco su un minicomputer e più esattamente su un TRS-80 con 16K di memoria. Il resto è storia. Scott Adams (questo era il suo nome) divenne il famoso (e ricco) autore di numerosi giochi di questo tipo, e fu seguito da frotte di imitatori più o meno bravi.

Vi proponiamo un listato-esempio per il Commodore 64.

	NORD	EST	SUD	OVEST
STANZA DEL MAGO	2	Ø	Ø	Ø
STANZA DEL GUARDIANO	3	Ø	1	4
STANZA DELLA SABBIA	Ø	Ø	2	Ø
CORRIDOIO	2	Ø	Ø	5
STANZA REALE	Ø	4	Ø	Ø

Figura 2

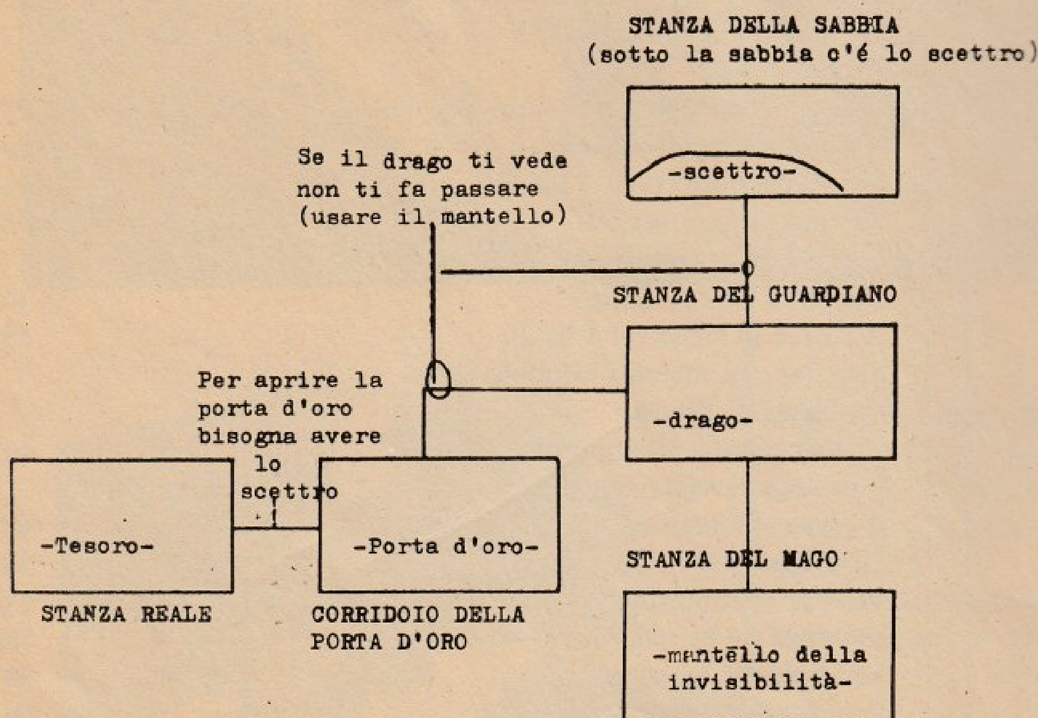


Figura 1

Come si gioca

Per giocare dovete dare istruzioni al computer sul da farsi, normalmente usando frasi di una o due parole, in inglese. Un comando tipico può essere «Go North», che dice al computer di andare a nord, e via di questo passo.

Mano a mano che si trovano oggetti si valuta se è il caso di prenderli o meno (ogni cosa, anche la più insignificante, ha il suo peso), considerando che solitamente non ci si può caricare più di un tanto. Il gioco è particolarmente interessante perché ci sono sempre da scoprire cose nuove, e i suggerimenti o indizi sul cosa fare sono disseminati ovunque. Una cosa è però sapere che questi suggerimenti esistono, altra è estrarli dalle parole che il computer vi dà, capire poi cosa vogliono dire è ancora un'altra cosa. Un altro motivo di successo è dovuto alle risposte che un gioco ben congegnato può fornire: si ha spesso l'impressione di parlare con una persona, e non con una stupida macchina senza cervello.

Veniamo ora alle note dolenti. Esistono numerosissimi giochi di questo genere, per tutti i tipi di Home e Personal Computer, ma sono per la maggior parte in inglese. Esiste anche qualche traduzione, ma niente, per quanto mi risulti al momento, tradotta in «italiano».

Questo articolo cercherà di spiegare tutti i trucchi (o almeno i più importanti) che devono essere usati per scrivere questo tipo di gioco. A corredo dell'articolo, come abbiamo già precisato, è presente un listato per C64

```
READY.

1 PRINT"£"
2 PRINT"SEI SU SIRIUS B, E DEVI SCOPRIRE
 IL      SEGRETO DELLA PIRAMIDE PER "
3 PRINT"RIUSCIRE A SCAPPARE DALLA BARRIE
RA       DIFENSIVA DEL PIANETA"
4 PRINT:PRINT"COMANDI UTILI: N,E,S,O PER
 LE DIREZIONI C    PER PULIRE LO SCHERMO"
5 PRINT"G          PER RIPETERE LA DESCRIZIO
NE       I         PER SAPERE COSA HAI"
6 PRINT"F          PER TERMINARE IL GIRO":PR
INT
7 REM xxxxxx INIZIALIZZAZIONE  xxxxxxx
8 DIMDS(16,4):P$="000010101011111111":DI
MF(6),O(18),O$(18),D$(16):LN=4
10 DIMNT$(2),NU(2),NU$(22,2):OL=18:S=2:M
X=3:NL=22
12 FORX=1TO18:READD$(X):NEXTX:FORX=0TO22
:FORY=0TO1:READNU$(X,Y)
13 NEXTY:NEXTX
15 FORX=1TO16:FORY=1TO4:READDS(X,Y):NEXT
Y:NEXTX
17 FORX=1TO18:READO$(X):NEXTX
18 FORX=1TO18:READO(X):NEXTX
26 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
27 REM xxxx  LETTURA INPUT    xxxxxxxxx
28 REM xxxx E TEST PER SITUAZIONI xxxx
29 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
30 GOSUB47
32 INPUT"COMANDO ";C$:PRINT:GOSUB4900:IF
SR=1 THEN SR=0:GOTO 1100
33 IF SR=2THENSR=0: GOTO 30
34 IFFTHENPRINT"USI PAROLE CHE NON CONOS
CO!!!":GOTO32
35 GOTO300
36 IFS=8ANDF(4)=0THENPRINT"C'E' UNA PICC
OLA FESSURA NELLA PARETE"
38 IFS=4ANDF(1)=-1THENPRINT"IL DRAGO DEC
IDE CHE NON GLI PIACI E TI  MANGIA":GOTO
8000
40 IFS=4ANDF(1)=0THENPRINT"IL DRAGO STA
BLOCCANDO UNA USCITA A EST":F(1)=-1
41 IFS<>4ANDF(1)=-1THENF(1)=0
42 IFS=4ANDF(1)=1THENPRINT"IL DRAGO E' M
ORTO"
43 IFS=16ANDF(2)=0THENPRINT"C'E' QUALCOS
A CHE LUCCICA SULLA CIMA    DELL'ALBERO.
44 IFS=16ANDF(2)=1THENPRINT"L'ALBERO E'
ADAGIATO SUL PAVIMENTO."
45 IF(S=12ANDF(3)=1)ORS=14THENPRINT"LA P
ORTA E' SFONDATA."
46 RETURN
47 K=-1:IFLEFT$(D$(S),1)="*"THENPRINTMID
$(D$(S),2)::GOTO52
48 REM xx STAMPA DESCRIZIONE STANZA xxx
49 REM xx E OGGETTI        xxxxxxxxxxxxxx
50 PRINT"SEI IN ";D$(S);
52 FORZ=0TOOL:IFKTHENIFO(Z)=STHENPRINT:P
RINT"COSE INTERESSANTI QUI :";K=0
54 GOTO70
56 TP$=O$(Z):IFRIGHT$(TP$,1)<>"/"THENRET
URN
58 FORW=LEN(TP$)-1TO1STEP-1:IFMID$(TP$,W
,1)="/"THENTP$=LEFT$(TP$,W-1):RETURN
60 NEXTW
62 RETURN
70 IFO(Z)<>STHEN74
72 GOSUB56:PRINT"UN ";TP$
74 NEXT Z:PRINT
80 K=-1: FORZ=0 TO 4: IFKTHENIFDS(S,Z)<>
0THENPRINT:PRINT"USCITE VISIBILI A: ";K=0
82 IFDS(S,Z)<>0THENPRINTNU$(Z,1);"  ";
84 NEXTZ:PRINT:PRINT:GOTO36
297 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
298 REM xxxx   GESTIONE VERBI    xxxxxxx
299 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
300 X=NU(0):IFX<2THENX=2
305 X=INT(X/2)
310 ONXGOTO400,650,700,750,800,850,900,9
50,1000,1050,1100
320 PRINT"NON HO CAPITO COSA VUOI FARE!!
":GOTO32
397 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
398 REM xxxx   GESTIONE DIREZIONI xxxxx
399 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
400 IFNU(1)<1THENPRINT"DAMMI ANCHE UNA D
IREZIONE.":GOTO32
420 IFDS(S,NU(1))=0THENGOTO3000
425 IF DS(S,NU(1))=-1THEN6000
430 S=DS(S,NU(1)):GOTO3005
647 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
648 REM xxxxxx  PRENDI   xxxxxxxxxxxxxx
649 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
650 IFO(NU(1))=-1THENGOTO3010
660 IFO(NU(1))<>STHENGOTO3015
665 IFIV>MXTHENGOTO3020
670 IFMID$(P$,NU(1),1)="0"THEN3030
675 O(NU(1))=-1:IV=IV+1:GOTO3040
697 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
698 REM xxxxxx POSA/GETTA   xxxxxxxxxxxx
699 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
700 IFO(NU(1))<>-1THENGOTO3035
710 O(NU(1))=S:IV=IV-1:IFNU(1)=13THENF(5
)=0
715 GOTO3040
747 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
748 REM xxxxxx TAGLIA/SEGA   xxxxxxxxxxx
749 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
750 IFNU(1)<>8ORF(2)=1THEN3030
760 IFO(7)<>-1ORO(11)<>-1THEN3030
765 IFS<>16THEN3015
770 IFF(6)=0THENPRINT"LA SEGA NON LAVORA
 SENZA CORRENTE!!!!":GOTO32
775 IFF(5)=0THENPRINT"L'ALBERO TI CADE I
N TESTA. CRUNCH.":GOTO8000
780 F(2)=1:PRINT"L'ALBERO CADE SUL TUO E
LMETTO. UN'ASCIA CADE DALL'ALBERO"
785 O(18)=5:GOTO3040
797 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
798 REM xxxxxx DISTRUGGI/ABBATTI  xxxxx
799 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
800 IFNU(1)<>10THEN3030
805 IFO(18)<>-1ORF(3)=1THEN3030
810 IFS<>12THEN3015
815 PRINT"CHOP CHOP CRUNCH FSSSTUMP!!!!
IL PORTONE E' STATO SFONDATO!!!!"
820 F(3)=1:DS(S,2)=14:GOTO3040
847 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
848 REM xxxxxx METTI/INDOSSA     xxxxxxx
849 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
850 IFNU(1)<>17THEN3030
855 IFO(17)<>-1THEN3035
860 F(5)=1:GOTO3040
897 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
898 REM xxxxx CONNETTI/COLLEGA  xxxxxxx
899 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
900 IFNU(1)<>11ANDNU(1)<>7THEN3030
905 IFO(NU(1))<>-1THEN3035
910 IFO(11)<>-1ORO(7)<>-1THEN3030
915 F(6)=1:GOTO3040
947 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
948 REM xxxxx INSERISCI/INFILA  xxxxxxx
949 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
950 IFNU(1)<>14THEN3030
955 IFO(NU(1))<>-1THEN3035
956 IFS<>8THENPRINT"NON VEDO DOVE POTERL
O INSERIRE!!":GOTO32
960 O(NU(1))=0
970 PRINT"LA PARETE DI OVEST SI SPOSTA!!
!          ORA C'E' UN PASSAGGIO!!!"
975 F(4)=1:DS(S,3)=10:GOTO3040
997 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
998 REM xxxxx AFFILA/ARROTA    xxxxxxxxx
999 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1000 IFNU(1)<>15THEN3030
1005 IFO(NU(1))<>-1THEN3035
1006 IF O(9)<>-1THENPRINT"NON HAI NULLA
CON CUI AFFILARLO!":GOTO32
1010 PRINT"IL COLTELLO DA BURRO DIVENTA
UN PUGNALE AFFILATISSIMO!!":O(15)=0
1020 O(16)=-1:GOTO3040
1047 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1048 REM xxxxx UCCIDI/ACCOLTELLA xxxxxxx
1049 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1050 IFNU(1)<>6ORO(16)<>-1THEN3030
1055 IFS<>4THEN3015
1060 IFF(1)=1THENPRINT"LA POVERA COSA E'
 GIA' MORTA.":GOTO32
1070 F(1)=1:PRINT"SQUELCH. IL PUGNALE AF
FONDA FINO "
1080 PRINT"ALL'IMPUGNATURA NEL CORPO DEL
 DRAGO.    E' MORTO. POVERA COSA"
1090 DS(S,4)=15:GOTO3040
1097 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1098 REM xxxxx INVENTARIO         xxxxxxx
1099 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1100 PRINT"£":PRINT"CON TE HAI:"
1110 NC=0:FORZ=1TO18:IFO(Z)=-1THENNC=NC+
1:GOSUB56:PRINT"UN ";TP$
1115 NEXTZ
1120 IFNC=0THENPRINT"NIENTE."
1125 PRINT:GOTO32
2997 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2998 REM xxxxx   MESSAGGI          xxxxxxx
2999 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
3000 PRINT"NON PUOI ANDARE IN QUESTA DIR
EZIONE!!":GOTO32
3005 PRINT"FATTO!":GOTO30
3010 PRINT"LO HAI GIA'!!":GOTO32
3015 PRINT"DOVE LO VEDI??!!":GOTO32
3020 PRINT"SEI TROPPO CARICO!!!!":GOTO32
3030 PRINT"NON PUOI, IDIOTA!!!":GOTO32
3035 PRINT"NON CE L'HAI, STUPIDO!!!":GOT
O32
3040 PRINT"FATTO!":GOTO32
4897 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
4898 REM xxxxx  ANALISI INPUT    xxxxxxx
4899 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
4900 IFLEN(C$)>1THEN5000
4910 IFC$="N"THENC$="VAI NORD":GOTO5000
4920 IFC$="E"THENC$="VAI EST ":GOTO5000
4930 IFC$="S"THENC$="VAI SUD ":GOTO5000
4940 IFC$="O"THENC$="VAI OVEST":GOTO5000
4950 IFC$="I"THEN SR=1:RETURN
4960 IFC$="C"THENPRINT"£":SR=2:RETURN
4970 IFC$="G"THENPRINT"£":SR=2:RETURN
4980 IFC$="F"THEN8010
5000 K=0:NT$(0)="":NT$(1)=""
5005 FORX=1TOLEN(C$):K$=MID$(C$,X,1):IFK
$=" "THENK=1:GOTO5015
5010 NT$(K)=LEFT$(NT$(K)+K$,LN)
5015 NEXTX
5020 FORX=0TO1:NU(X)=0:IFNT$(X)=""THEN50
50
5025 FORY=0TONL:K$=NU$(Y,X):IFLEFT$(K$,1
)="*"THENK$=MID$(K$,2)
5030 IFX=1ANDY<5THENK$=LEFT$(K$,LN)
5035 IFNT$(X)=K$THENNU(X)=Y:GOTO5045
5040 NEXTY:GOTO5050
5045 IFLEFT$(NU$(NU(X),X),1)="*"THENNU(X
)=NU(X)-1:GOTO5045
5050 NEXTX:IF(NU(0)=5ORNU(0)=6)ANDNU(1)<
1THENF=0:RETURN
5060 F=NU(0)<1ORNU(1)<1:RETURN
5997 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
5998 REM xxxxx  STANZA -1          xxxxxxx
5999 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
6000 PRINT"£":PRINT"RAGGI LASER ESCONO D
ALLE ARMI FISSATE ALLA STRADA"
6010 IFO(12)<>-1THEN6050
6020 IFO(13)<>-1THEN6080
6030 PRINT"ENTRAMBI I RAGGI SONO RESPINT
I DALLO     SPECCHIO E DAL RIFLETTORE!!!"
6035 PRINT"SEI RIUSCITO A SCAPPARE VIVO!
!          CONGRATULAZIONI!!!":GOTO8010
6050 PRINT"IL RAGGIO PROVENIENTE DA SINI
STRA E'    RESPINTO DALLO SPECCHIO."
6055 PRINT"QUELLO DI DESTRA NO!!!":GOTO8
000
6080 PRINT"IL RAGGIO PROVENIENTE DA DEST
RA E'       RESPINTO DAL RIFLETTORE."
6085 PRINT"QUELLO DI SINISTRA NO!!!":GOT
O8000
7997 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
7998 REM xxxxx  FINE GIOCO        xxxxxxx
7999 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
8000 PRINT"SEI MORTO!"
8010 PRINT"UN ALTRO 'GIRO'? ";:INPUTY$:IF
LEFT$(Y$,1)="S"THENRUN
8020 END
9007 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
9008 REM xxxxx DESCRIZIONI STANZE xxxxxx
9009 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
9010 DATA"UNA STRADA. DUE PEZZI METALLIC
I PUNTANO A OVEST"
9020 DATA"UNA CURVA DI UNA LARGA STRADA"
9030 DATA"UNA PICCOLA E SCURA BARACCA"
9040 DATA"UNA PICCOLA E SCURA BARACCA"
9050 DATA"UNA STRADA CHE VA DA NORD A SU
D. CI SONO BARACCHE AI LATI"
9060 DATA"UNA STRADA CHE VA A NORD. C'E'
  UNA              PIRAMIDE A SUD"
```

adattabile facilmente a qualunque computer.

Se vi cimenterete in questo lavoro, scrivendo la «vostra» avventura, vi accorgerete che a fine lavoro avrete incrementato in maniera considerevole la vostra conoscenza del Basic. E questo per una ragione molto semplice: nei programmi di tipo «Adventure» si fa un largo uso di tutte le possibilità del Basic: manipolazione delle stringhe alfanumeriche e delle variabili in maniera molto spinta, generazione dei numeri casuali, logica degli If... THEN, LOOP, presentazione dello schermo, concatenamenti e, se il vostro computer ha una seppur limitata capacità di grafica, colore e suoni, vi potete spingere alla ottimizzazione di semplici effetti sonori e/o grafici. Il tutto si applica ovviamente anche alla programmazione in linguaggio macchina.

Pianificazione dell'avventura

Prima di cominciare a scrivere il programma dovete rispondere alle seguenti domande:

1) Dove si svolge l'Avventura?

2) Che tipo di mondo si vedrà (reale, fantastico, storico...)?

3) Qual è lo scopo principale dell'avventura?

4) Che tipo di ostacoli dovranno essere superati e come?

Date queste risposte potete cominciare a fare un disegno della rete di «stanze» (d'ora in avanti userò questo termine per indicare qualunque località, sia essa collina, ruscello, antro o altro) della vostra avventura. Il disegno dovrebbe risultare simile a quello mostrato (solo in piccola parte) in **figura 1**.

Così, per trovare il tesoro bisogna aprire la porta d'oro; per aprire la porta d'oro bisogna avere lo scettro e passare sotto il naso del drago protetti dal mantello dell'invisibilità, mentre per trovare lo scettro bisogna scavare nella sabbia. Una volta che avete risposto alle domande e disegnato la mappa della vostra avventura siete pronti a cominciare a scrivere il vostro programma.

Il programma

Il programma per gestire questo tipo di gioco è diviso in due parti: il programma vero e proprio che gestisce la logica del tutto e il Data Base, e cioè l'insieme dei dati su cui lavorerà il programma. La parte più importante è appunto il Data Base, in quanto, se il gioco è ben congegnato, cambiando il Data Base si cambierà avventura. Non è facile ottenere questo risultato ma è possibile ed è il sistema usato da molti programmatori di questo tipo di gioco. Vediamo ora separatamente le due parti.

Il data base

Non preoccupatevi del termine tecnico, in pratica dovrete scrivere una lista di lunghezza variabile di istruzioni Data, che conterranno descrizioni, numeri e parole. I modi in cui potete studiare questo insieme di Data sono molti, e qui ne vedremo alcuni, ma l'importante è capire il modo in cui questi dati vengono usati; fatto questo potrete scrivere il tutto a modo vostro.

Il Data Base di ogni avventura, in qualunque modo sia gestito, deve comprendere almeno quattro tabelle (di cui qualcuna «doppia», con una parte numerica e una alfanumerica); il **Vocabolario**, gli **Oggetti**, le **Stanze** e i **Messaggi**. Se si va un po' più nel complesso si può avere anche la tabella dei comandi. Vediamo tutte queste tabelle nel dettaglio.

Vocabolario

Questa tabella (o matrice se preferite) è divisa in due parti: **verbi** e **nomi**. Contiene tutte le parole che l'Avventura può gestire (verbi, oggetti, azioni etc.). I sinonimi (dei verbi) sono gestiti.in diversi modi, il più diffuso è quello di far precedere al sinonimo un asterisco (*).

Esempio:

Vai, *cammina, *corri, mangia, *assaggia.

Ovviamente in questo esempio ha assunto che i vari verbi di movimento siano gestiti dal programma tutti alla stessa maniera.

Quando girerà il programma, ci sarà una subroutine che confronterà le due (o una) parole dell'INPUT con questa tabella, e darà come risultato uno o due numeri, identificanti la posizione del verbo e/o del nome nella matrice. Dopodiché la maggior parte del programma giocherà su questi due numeri. Ogni verbo avrà una routine di gestione appropriata, che sarà chiamata con un ON x GOTO, (o, sui personal con la gestione del GOTO a variabili, con un GOTO x). La routine così chiamata farà per prima cosa un controllo sulla «sensatezza» dell'azione (es: MANGIA SEDIA produrrà un bel «Non essere ridicolo» o cose del genere); poi si preoccuperà della gestione del verbo col nome eventualmente associato.

Oggetti

Questa tabella contiene tutte le informazioni necessarie per gestire i vari oggetti che saranno trovati dal giocatore. È la tabella più modificata nel corso del gioco, perchè tutte le volte che il giocatore prenderà o poserà un oggetto, qualcosa viene cambiato in questa matrice. Questa tabella è divisa in due parti: **numero stanza** e **descrizione.** La descrizione contiene appunto la descrizione dell'oggetto. Il numero stanza è invece una tabella numerica contenente il numero della stanza in cui si trova l'oggetto in questo momento. Se il valore della stanza è zero, vuol dire che l'oggetto non è da nessuna parte (per ora, potrebbe comparire in seguito a certe azioni del giocatore). Se il valore della stanza è meno uno (-1), vuol dire che l'oggetto è in mano al giocatore. Fin qui tutto semplice (spero): quando si vorrà stampare la lista degli oggetti presenti in una stanza, basterà confrontare il numero della stanza in cui si trova il giocatore con il numero contenuto nella parte numerica della tabella degli oggetti. Se il numero è lo stesso, basterà stampare la descrizione dell'oggetto presa dalla corrispettiva entrata delle descrizioni della tabella oggetti. Un esempio delle due parti della tabella oggetti potrebbe essere il seguente:

Descrizioni: Ascia, Penna, Bottiglia vuota, Bottiglia piena

Numero stanza: - 1, 2, 5, 0

In questo esempio l'ascia è in mano al giocatore; nella stanza numero cinque c'è una bottiglia vuota, mentre la penna è nella stanza numero due. La bottiglia di acqua piena non è da nessuna parte. Quando il giocatore avrà trovato dell'acqua con cui riempire la bottiglia, il numero di stanza sarà invertito tra bottiglia vuota e bottiglia piena.

Ci sono altri due problemi da risolvere che riguardano gli oggetti: uno è la descrizione stessa, che quando viene data dovrebbe essere lunga (es.: vecchia ascia arrugginita), ma quando il giocatore vuole fare qualcosa con l'oggetto, deve essere breve (es.: prendi ascia, e non prendi vecchia ascia arrugginita).

Vediamo due soluzioni possibili al problema: una è quella di «spaccare» la descrizione in due parti. In questo caso si potranno riempire due elementi della matrice per ogni oggetto (es.: «ascia» e «arrugginita») e stamparli entrambi quando si dà la descrizione dell'oggetto, mentre si può «testare» solo il primo elemento quando si controlla l'INPUT del giocatore. L'altro sistema prevede che sia un

elemento solo a contenere sia la descrizione «lunga» che quella breve, separate da uno slash («/»). In questo caso la tabella descrizioni avrà un solo elemento per ogni oggetto (es: «vecchia ascia arrugginita/ascia/»; lo slash finale serve a capire subito che abbiamo questo tipo di descrizione. Questo sistema l'ho usato nel listato (vedi linee 9390 - 9460 per la lista degli oggetti e linee 56-74 per la stampa degli oggetti senza lo «/»).
L'altro problema che riguarda sempre gli oggetti è il controllo sul fatto se l'oggetto si può veramente prendere o posare (per esempio è meglio evitare che il giocatore possa prendere un albero). Uno dei modi per fare ciò è quello di mettere in una variabile alfanumerica degli «0» se l'oggetto non si può prendere, e degli «1» se l'oggetto relativo si può prendere (se abbiamo dieci oggetti ci sarà una serie di dieci caratteri che saranno o zero o uno a seconda dei casi). Un altro modo è quello di mettere nella tabella comandi (che vedremo più avanti) solo la gestione degli oggetti «prendibili», ma questo sistema è un po' più complesso.

Stanze

La matrice delle stanze contiene tutte le informazioni per ogni «stanza» in cui il giocatore può trovarsi. È anche questa divisa in due tabelle: **Descrizione** e **Direzioni.** La tabella delle descrizioni contiene la descrizione di ogni stanza, senza contenere, molto spesso, l'inizio della frase «Sei in», che sarà invece stampato davanti ad ogni descrizione (questo, ovviamente, per risparmiare memoria). Se poi si volesse, per qualche descrizione, evitare di stampare il «Sei in», si potrebbe mettere un asterisco in fronte alla descrizione, che farà in modo, opportunamente testato, di non stampare la parte di frase succitata. La lista delle descrizioni comincia alla linea 9010 e la stampa viene fatta alle linee 47-50, testando per l'asterisco.
La tabella delle direzioni è numerica, e contiene una entrata per ogni possibile direzione (per ogni stanza). Le direzioni possono essere quattro, otto o dieci: nord, nord est, est, sud est, sud, sud ovest, ovest, su, giù. Nei listati di esempio sono state considerate solo le quattro direzioni fondamentali. Ogni entrata della tabella consiste nel numero della stanza in cui si troverà il giocatore muovendo in quella direzione. Se c'è uno zero vuol dire che non c'è uscita in quella direzione. Nella **figura due,** se siamo nella stanza numero 2 (la stanza del guardiano) e ci muoviamo a nord, ci troveremo nella stanza della sabbia (stanza numero 3), se invece volessimo andare a sud, ci troveremo nella stanza del mago, e così via. Nel listato a 9300 si trovano le direzioni. Da notare che questa tabella viene usata anche per stampare le uscite visibili, dopo la stampa della descrizione della stanza (80-84 da notare che la stampa delle direzioni viene fatta utilizzando la matrice del vocabolario).

Messaggi

Mettere tutti i messaggi (o almeno la gran parte) che possono essere mostrati al giocatore in una tabella, ha senso specialmente se si usa anche la tabella comandi (vedremo il perché). Comunque nella vostra avventura ci dovranno essere per forza dei messaggi (tipo: «Non succede nulla», oppure «Non puoi!!!» e così via). Nelle avventure più semplici dovendo stampare un messaggio si fa un GOTO ad una linea di programma che stampa il messaggio e ritorna dove deve (il più delle volte ad accettare un altro comando). Questo sistema è stato usato in entrambi i listati (vedere alla linea 3000). Per la gestione dei messaggi in matrice tenete presente, per ora, che dovrete avere una matrice alfabetica contenente tutti i messaggi, e che ovviamente ogni messaggio avrà un numero che lo identifica (il numero di elemento della matrice).

Monitor in linguaggio macchina per VIC20

di Eugenio Rapella

Questo interessante programma, realizzato da **Eugenio Rapella,** è ben fatto e molto utile per la «gestione» di programmi in linguaggio macchina, che stanno prendendo sempre più piede, soprattutto per i VIC-20 che non hanno espansioni di memoria. Caricato il programma Basic, se non si hanno espansioni di memoria, si ha qualcosa come 679 bytes disponibili al linguaggio macchina, che permettono, pur nella loro esiguità, di fare tante belle cosucce. L'autore si è prodigato molto (e lo vedrete fra poco) nello spiegare come si usa il suo programma con tutti i suoi innumerevoli comandi. Vediamo qualche piccola lacuna nei commenti al listato (i REMarks, per intenderci), che sono presenti, ma non abbondanti né molto dettagliati.
L'idea non è nuova, ma lo sviluppo è molto originale, come pure i comandi che sono a disposizione dell'utente. Detto questo non ci resta che passare la parola al diretto interessato.
La «gestione» (caricamento, esecuzione,...) di un programma in linguaggio macchina (L.M.) per il 6502 attraverso il VIC 20 è, se non si dispone dell'apposita cartridge, piuttosto difficoltosa. Questo «Monitor L.M.», realizzato per un VIC 20 versione standard, può essere un valido aiuto per chi si accinge alle prime esperienze in questo senso. Effettuato il caricamento e l'esecuzione del programma Basic, viene presentato il «menù» e la «locazione di riferimento in memoria» (il cui valore è stampato in reverse) che, inizialmente, è la 7000 (d'ora in poi il termine «locazione di riferimento» verrà abbreviata con LR).
A questo punto il programma attende:
1) l'inserimento del codice esadecimale della prima istruzione del programma in L.M. che verrà caricata nella LR.
oppure
2) la pressione di uno dei tasti descritti nel «menù» con le relative conseguenze.
L'inserimento di codici differenti da quelli previsti provoca la stampa di un doppio punto interrogativo. Nel primo caso (ad esempio digitando A9), il codice esadecimale viene trasformato in decimale e caricato nella LR; sullo schermo appaiono, sulla stessa riga, i seguenti quattro valori, come in **figura 1.**
Viene quindi stampata la successiva LR in attesa della prossima istruzione. Prima di inserire i codici esadecimali del programma in L.M., è preferibile cancellare il contenuto della memoria ad esso riservato, tramite una delle opzioni disponibili (N=NEW LM).
Vediamo in dettaglio le possibilità offerte dal «menù» (caso 2):
Pressione del tasto:
T = DEC > ESA consente la conversione di un valore decimale in esadecimale.
es. DEC = ? 78
ESA = 4 E
H = ESA > DEC La LR non viene modificata. Consente la conversione da esadecimale a decimale.
es. ESA = ? 1F
DEC = 31
b = LIST PA La LR non viene modificata. La pressione della barra spaziatrice visualizza il contenuto della LR come da figura 1. La LR viene incrementata di 1.
↑ = LOC. PRE visualizza il contenuto di LR - 1 (come da figura 1) che diviene la

nuova LR.

U = ULTIME visualizza il contenuto delle ultime locazioni di memoria utilizzate nel programma L.M., ovvero quelle il cui contenuto è diverso da zero. Il numero dell'ultima memoria utilizzata è indispensabile per la registrazione di un programma in L.M. su cassetta.

R = RUN L.M. manda in esecuzione il programma L.M. Viene richiesto il valore decimale della locazione di «inizio esecuzione» in modo da consentire l'esecuzione di segmenti del programma L.M. redatto o l'esecuzione di un programma caricato in altra zona. Il valore iniziale normale è 7000. Terminata l'esecuzione del programma L.M., la pressione di un tasto qualsiasi consente il rientro al Basic.

L = LOAD LM consente il caricamento di un programma in L.M. da cassetta (opportunamente registrato tramite l'opzione «S»).
Viene richiesto il valore decimale dell'area di memoria a partire da cui il programma L.M. deve essere caricato: il valore normale è 7000. A caricamento effettuato, viene stampato il nome del programma; la LR viene posta a 7000.

S = SAVE LM consente la registrazione su cassetta di un programma in L.M. presente in memoria dalla locazione 7000 fino all'ultima utilizzata (riscontrabile tramite l'opzione «U») il cui valore viene richiesto. Viene anche richiesto un nome di riferimento (TITOLO?) a cui viene automaticamente posto il suffisso «LM» in reverse.

L = LIST... consente di visualizzare i codici macchina contenuti dalla locazione «x»: LOC IN. (DEC) (ovvero «numero della locazione iniziale espresso in decimale») alla locazione «y»: LOC FIN(DEC) (ovvero «numero della locazione finale espressa in decimale»). Deve essere 7000 ≤ x ≤ y ≤ 7679; un'errata impostazione dei valori provoca un

Listato - Monitor L.M.

```
1 POKE55,88:POKE56,27:CLR:POKE36879,110:PRINT"#":P=7000:U=7679:C=P:GOTO100
10 H$=""
20 IFDTHENA=INT(D/16):H$=MID$("0123456789ABCDEF",1+D-A*16,1)+H$:D=A:GOTO20
30 RETURN
32 GETR$:IFR$=""THEN32
33 RETURN
40 D=0:IFH$>""THENFORI=1TOLEN(H$):A=ASC(MID$(H$,I,1))-48:D=D*16+A+(A>9)*7:NEXT
42 RETURN
44 F1=0
46 IFABS(INT(X1))=X1ANDX1>=PTHEN50
48 PRINT"X*1'DATO NON VALIDO":F1=1:RETURN
50 IFABS(INT(X2))=X2ANDX2<=UTHEN54
52 PRINT"X*2'DATO NON VALIDO":F1=1:RETURN
54 IFX1<=X2THENRETURN
56 PRINT"X*1'<2' !*":F1=1:RETURN
60 PRINT"XT=DEC>ESA|H=ESA>DEC"
62 PRINT" =LIST PA|↑=LOC.PRE"
64 PRINT"U=ULTIME |R=RUN L.M"
66 PRINT"L=LOAD LM|S=SAVE LM"
68 PRINT"£=LIST...|/=DEL ..."
70 PRINT"I=INST...|N=NEW LM"
72 PRINT"*=BASIC  |M=MENU'"
73 PRINT"←=INIZIO |@=CM DEC"
74 PRINT"----------------------"
75 RETURN
82 IFCD<48ORCD>70ORCD>57ANDCD<65THENPRINT"??":GOTO120
84 RETURN
90 R=PEEK(LC):D=R:GOSUB10:A$=H$:D=LC:GOSUB10
93 IFA$=""THENA$="00"
94 PRINTA$;TAB(3);RSPC(5-LEN(STR$(R)))LC;TAB(3);H$
96 RETURN
100 PRINT"J* MONITOR L.M. *"
102 PRINT"*                *"
104 PRINT"* E.RAPELLA 83 *X":GOSUB60
120 PRINTTAB(9)"#"C
125 GOSUB32
130 V=ASC(R$)
135 IFV=32THENLC=C:GOSUB90:C=C+1:GOTO120
137 IFV=94THENC=C-1:LC=C:GOSUB90:GOTO120
140 IFV<>77THEN155
150 GOSUB60:GOTO120
155 IFV<>95THEN160
157 C=7000:GOTO120
160 IFV<>84THEN190
170 INPUT"DEC=";D
180 GOSUB10:PRINT"ESA="H$:GOTO120
190 IFV<>72THEN212
200 INPUT"ESA=";H$
210 GOSUB40:PRINT"DEC="D:GOTO120
212 IFV<>64THEN220
214 INPUT"C.M. IN DEC";D
216 POKEC,D:LC=C:GOSUB90:C=C+1:GOTO120
220 IFV<>42THEN240
230 PRINT"XRIENTRO AL BASIC":END
240 IFV<>78THEN255
242 PRINT"XCANCELLO PRG.LM        CONFERMI [S/N]"
245 GOSUB32:IFR$="S"THEN250
248 GOTO120
250 FORX=PTOU:POKEX,0:NEXT:RUN
255 IFV<>85THEN320
260 FORH=UTOPSTEP-1
270 IFPEEK(H)<>0THEN290
280 NEXT
290 IFH<P+3THENH=P+3
300 FORJ=H-3TOH+3:LC=J:GOSUB90:NEXT:GOTO120
320 IFV<>82THEN390
330 PRINT"JESECUZIONE PRG.L.M"
340 PRINT"XSCRIVI LOC D'INIZIO"
350 INPUT"XIN DEC [7000]";E
360 IFE<PORE>UTHENPRINT"?":GOTO340
380 PRINT"J":SYSE
382 GOSUB32:END
390 IFV<>92THEN440
400 INPUT"XLOC IN.(DEC)";X
410 INPUT"XLOC FIN(DEC)";Y
420 X1=X:X2=Y:GOSUB44:IFF1THEN400
430 FORH=XTOY:LC=H:GOSUB90:NEXT:C=Y+1:GOTO120
440 IFV<>47THEN550
450 PRINT"XELIMINO LM [X,Y] ES.IN"
470 INPUT"LOC X [DEC]";X
480 INPUT"LOC Y [DEC]";Y
490 X1=X:X2=Y:GOSUB44:IFF1THEN450
500 FORT=Y+1TOU
510 POKET-Y+X-1,PEEK(T):NEXT
520 FORT=U-Y+XTOU:POKET,0:NEXT
540 FORT=X-1TOX+1:LC=T:GOSUB90:NEXT:C=X:GOTO120
550 IFV<>73THEN670
560 PRINT"XCREA SPAZIO PER'I'INS"
570 PRINT"A PARTIRE DA LOC 'M+1'"
590 INPUT"LOC [M]";M
600 INPUT"QUANTI INST";I
```

Listato - Monitor L.M.

```
610 IFI=0THEN120
620 IFM+I>UTHENPRINT"*IMPOSSIBILE [U]*":GOTO120
630 FORT=U-ITOM+1STEP-1
640 POKET+I,PEEK(T):NEXT
650 FORT=M+1TOM+I:POKET,0:NEXT
660 FORT=MTOM+I+2:LC=T:GOSUB90:NEXT:C=M+1:GOTO120
670 IFV<>83THEN730
680 PRINT"■REGISTRAZIONE LM"
690 PRINT"■ULTIMA MEMORIA"
692 PRINT"UTILIZZATA NEL PRG.LM"
694 INPUT"[DEC]",W
696 INPUT"TITOLO";T$
697 T$=T$+"■LM■"
700 OPEN1,1,2,T$
705 PRINT#1,T$
710 FORT=PTOW:PRINT#1,PEEK(T):NEXT
720 CLOSE1:PRINT"■OK, FATTO":C=P:GOTO120
730 IFV<>76THEN800
740 PRINT"■CARICAMENTO LM"
745 PRINT"■DA CASSETTA■■"
750 INPUT"1^LOC [7000]";W
760 OPEN1,1,0,T$
765 INPUT#1,T$
770 INPUT#1,VX:POKEW,VX
780 IFST=64THENCLOSE1:PRINT"CARICATO ";T$:C=P:GOTO120
790 W=W+1:GOTO770
800 CD=V:GOSUB82:PRINTR$;
810 GETS$:IFS$=""THEN810
820 PRINTS$:CD=ASC(S$):GOSUB82:H$=R$+S$:GOSUB40
830 POKEC,D:LC=C:GOSUB90:C=C+1:GOTO120

READY.
```

Commento al listato «Monitor L.M.»

Istruz. n. 1	Protegge l'area di memoria da 7000 (P) a 7679 (U) dal Basic; determina la colorazione di sfondo e il colore del carattere di stampa.
10-30	Subroutine di trasformazione dal decimale D all'esadecimale H$.
32-33	Subroutine d'attesa.
40-42	Subroutine di trasformazione dell'esadecimale H$ nel decimale D.
44-56	Subroutine di controllo di validità dati: se $P \leq X1 \leq X2 \leq U$ è $F1 = 0$ (tutto bene), in caso contrario è $F1 = 1$ (errore)
60-75	Subroutine: visualizza il menù ovvero l'elenco dei tasti associati alle varie operazioni.
82-84	Subroutine: controlla che il tasto premuto sia uno di quelli previsti (codice ESA / tasto «menù»).
90-96	Subroutine di stampa: in riferimento, alla locazione LC, calcola e visualizza i valori di fig. 1.
100	Inizio del programma principale.
120	C'è la locazione di riferimento (LR)
135	Gestisce l'opzione «/» (barra spaziatrice)
137	Gestisce l'opzione ■↑■
140-150	Gestisce l'opzione «M»
155-157	Gestisce l'opzione ■←■
160-180	Gestisce l'opzione «T»
190-210	Gestisce l'opzione «H»
212-216	Gestisce l'opzione « »
220-230	Gestisce l'opzione «*»
240-250	Gestisce l'opzione «N»
255-300	Gestisce l'opzione «U»
320-382	Gestisce l'opzione «R»
390-430	Gestisce l'opzione «L»
440-540	Gestisce l'opzione «/»
550-660	Gestisce l'opzione «|»

(Se non vi è spazio sufficiente per l'inserimento richiesto, viene visualizzato il messaggio: «* IMPOSSIBILE (U) *» con cui si ricorda che, con l'opzione «U», conviene verificare le ultime locazioni utilizzate dal programma L.M.).

670-720	Gestisce l'opzione «S»
730-790	Gestisce l'opzione «L»
800-830	Accetta i codici esadecimali delle istruzioni del programma in linguaggio macchina.

messaggio appropriato. La LR diviene Y + 1.

/ = DEL cancella i codici macchina dalle locazioni «x» a «y» comprese traslando i codici successivi a «y». La «coda finale» viene riempita con valori nulli. Come da esempio in **figura 2** alla fine dell'operazione vengono visualizzati i contenuti di «y - 1», «y», «y + 1». La LR viene posta uguale a «y».

I = INST consente l'inserimento di nuove istruzioni esadecimali tra quelle già caricate in memoria creando l'opportuno «spazio» come nell'esempio in **figura 3**.

Ad operazione avvenuta vengono visualizzati i contenuti delle locazioni in prossimità di quelle da inserire. La LR viene modificata, nell'esempio LR diviene 7002 in attesa delle 3 nuove istruzioni da inserire.

N = NEW LM azzera i contenuti della memoria riservata al programma L.M. ovvero dalla locazione 7000 alla 7679. Questa operazione va sempre eseguita dopo il caricamento del programma Monitor L.M. quando si intenda redarre un programma L.M. Per evitare pressioni accidentali di questo tasto, viene richiesta una conferma prima di passare all'esecuzione.

*** = BASIC** termina il programma Monitor L.M. e rientra al Basic.

M = MENÙ ricorda i tasti associati alle varie operazioni.

← = INIZIO porta la LR al valore 7000.

CM DEC consente l'inserimento del codice di una istruzione L.M. in decimale anziché esadecimale.

Esempio:
C.M. IN DEC ? 141
Effetto:
8D 141 7001 1B59

Questa operazione è particolarmente utile nel caricamento di un programma L.M. già «decimalizzato» come routine in L.M. di un programma redatto in Basic. L'istruzione n. 1 del programma Basic modifica il

Monitor in L.M.

1) Codice esadecimale dell'istruzione	(es.: A9)
2) Codice decimale dell'istruzione	(es.: 169)
3) Locazione in decimale in cui il codice è stato inserito	(es.: 7000)
4) Locazione, in esadecimale, in cui il codice è stato inserito	(es.: 1B58)

Figura 1

Monitor in L.M.

Situazione

codice ESA	locazione
A9	7000
FF	7001
8D	7002
40	7003
0	7004
0	7005

impostando:
LOC IN. (DEC) = 7001
LOC FIN (DEC) = 7002

Effetto

codice ESA	locazione
A9	7000
40	7001
0	7002
0	7003
0	7004
0	7005

Figura 2

Situazione:

A9	7000
FF	7001
8D	7002
40	7003
0	7004
...	...

LOC (M) ? 7001
QUANTI INSERIMENTI ? 3
Nel rettangolo le locazioni visualizzate.

Effetto

A9	7000
FF	7001
00	7002 ← LR
00	7003
00	7004
8D	7005
40	7006
0	7007
...	...

Figura 3

puntatore del «limite della memoria» (locazioni 55 e 56), proteggendo l'area di memoria che va dalla locazione 7000 alla 7679 che risulta quindi disponibile per la memorizzazione delle istruzioni del programma in L.M. Nella realizzazione di questo programma per un VIC versione standard (3,5 K bytes), si è posto il problema di equilibrare la lunghezza del programma Basic con lo spazio disponibile per il codice macchina. Per evitare un'eccessiva occupazione di memoria del programma Basic, non è stato realizzato un controllo completo dei dati in «input», è opportuno comunque ricordare che qualsiasi segnalazione d'errore (come ILLEGAL QUANTITY ERROR...) non pregiudica il contenuto dell'area riservata al programma L.M., che risulta disponibile rimandando in esecuzione il programma MONITOR L.M. In altri termini, durante la redazione di un programma in L.M., è sempre possibile intervenire con il tasto RESTORE senza perdere i codici L.M. già redatti.

È ovvio come, disponendo di una qualsiasi espansione di memoria, sia possibile, modificando i valori nella prima istruzione, migliorare il programma Basic e/o aumentare lo spazio a disposizione per il programma L.M.

Il seguente programma in L.M. (da «VIC REVEALED» di Nick Hampshire, pag. 37) può essere utilizzato come prova su un VIC 20 versione standard:
(0 = zero)

A9 FF 8D 40 03 A2 FF AD 40 03 9D 00 1E
A9 04 9D 00 96 CE 40 03 CA D0 EF 60

il programma realizza la visualizzazione del set di caratteri del VIC 20.

Come descritto in precedenza, è possibile redarre un programma L.M. dalla locazione 7000, registrarlo su nastro e ricaricarlo in una diversa area di memoria. Ovviamente tale operazione è eseguibile per programmi che non contengono salti ad indirizzi assoluti del programma stesso.

È anche ovvio che l'esecuzione di un programma in L.M. si può ottenere rientrando al Basic e digitando l'opportuna istruzione SYS (es. SYS 7000).

Sprite Editor per gestire la grafica

di Walter Pistarini

Il COMMODORE 64 ha una delle migliori capacità grafiche degli Home Computer sul mercato, «battezzata» **Sprite.**

Che cos'è uno Sprite? Il nome, di per sè, non significa molto, esattamente Spirito, Folletto; ma il concetto è simile a quello della gestione grafica dei missili sugli ATARI.

Ogni Sprite è una entità ad alta risoluzione, costituita da un insieme di 21 righe di 24 punti ciascuna, ed ogni Sprite (fino ad un massimo di 8 contemporaneamente) è gestito dall'integrato che si cura di tutta la gestione video, chiamato VIC II (Video Interface Chip II). Per programmarne uno, tutto quello che dovete fare è di definire la sua sequenza di bytes, selezionare il suo colore, scegliere la sua posizione (su un asse orizzontale X che va da 0 a 512 e su un asse Y che va da 0 a 256) ed infine «accenderlo». Cambiando i valori di X ed Y, potete muovere lo Sprite in qualunque posizione dello schermo.

Ma veniamo ora a qualche dettaglio: come ho già detto, fino a otto Sprites possono essere mostrati sullo schermo contemporaneamente.

Ogni Sprite ha un puntatore (lungo 1 byte) sopra il blocco della memoria video. Il puntatore indica un blocco di 64 bytes entro un «banco» di 16K selezionato dal VIC II (voglio dire che il VIC II «vede» solo 16K, e in questi 16K ci devono stare la memoria video, i puntatori agli Sprites e i dati necessari per costruire gli Sprites).

L'ultimo byte dei 64 usati per descrivere uno Sprite è un byte di controllo; gli altri 63 contengono delle informazioni sui «punti» (o «Pixels») che costituiscono lo Sprite. Ogni gruppo di tre bytes rappresenta una riga di 24 punti dello Sprite. Nel modo standard, un bit (ce ne sono otto in un byte) messo a 1 mostra un punto del colore selezionato, mentre un byte messo a 0 mostra cosa c'è sotto di esso, (normalmente lo sfondo, ma potrebbe essere una parte di uno Sprite di più bassa priorità).

Associato con ogni Sprite ci sono alcune altre locazioni di memoria nel chip del VIC II. Il registro di abilitazione della visualizzazione ha un bit per ogni Sprite, come pure il registro di abilitazione al multicolore, quello di raddoppio della dimensione orizzontale e quello per la dimensione verticale; c'è poi quello della priorità dello Sprite rispetto allo sfondo, quello che si accorge della collisione fra due Sprites, e quello della collisione dello Sprite con lo sfondo.

Ancora, c'è un byte per ogni posizione verticale dello Sprite, ed un byte per il posizionamento orizzontale. Dato che ci sono più di 256 posizioni orizzontali, c'è anche un byte contenente un bit per ogni Sprite e che serve

Remarks Listato - Sprite Editor

10 Mette lo «Sfondo 0» in blu, dimensiona le matrici e inizializza le variabili X, Y, R, C, S ed S1.
20 Inizializza il VIC II mettendo a zero i vari registri di visualizzazione e raddoppio dello SPRITE, fa un RESTORE (questo perché potremmo arrivare qui anche per resettare il tutto, vedi riga 470, quando si vuole un nuovo schermo) e legge, nella matrice A$ i nomi di tutti i colori disponibili che saranno poi stampati sulla parte sinistra dello schermo.
30 Riempie tutta la matrice A con dei punti (valore ASCII del punto = 46), questo per riempire la griglia che sarà poi mostrata sullo schermo. Dopodiché riempie tutta la matrice B di zeri, perché, ovviamente, i DATA per costruire in seguito lo SPRITE a questo punto devono essere tutti a zero.
40 Posiziona lo SPRITE N. 2 sullo schermo, con coordinata X=60 e Y=200. Mette a posto il puntatore alle informazioni per costruire lo SPRITE N. 2 (puntatore che si trova a 2042) e gli dice di leggere i 63 bytes necessari all'indirizzo 832. Mette a posto i colori nei registri appropriati, utilizzando i colori che assume alla partenza (nero, blu chiaro, bianco) e che possono essere cambiati in qualunque momento con i tasti F3, F5 ed F7. Da notare che questa riga di programma (come molte altre) serve solo a poter mostrare lo SPRITE nella parte in basso a sinistra dello schermo.
50 Riempie con i dati dello SPIRITE (che si trovano nella matrice B), le locazioni di memoria usate dal VIC II per costruire lo SPRITE. Normalmente a questo punto le informazioni sono tutte a zero e non si vedrà nulla, ma se noi leggessimo da cassetta uno SPRITE precedentemente costruito, arrivando qui la matrice B conterrebbe valori «reali». Il secondo POKE serve a visualizzare lo SPRITE ed il terzo a dire al VIC II che lo SPRITE è di tipo multicolore.
60-125 Stampa la lista dei comandi disponibili.
130 Disegna la griglia di punti sullo schermo. Questo viene fatto stampando tutta la matrice A nella parte destra dello schermo.
140 Dopo aver messo X e Y a 1 va a 290 per disegnare il cursore.
150 Legge il comando battuto.
160 Mette il valore corrente letto nella matrice A dentro la griglia
170-185 Spostamenti del cursore: l'unica cosa che fa è di incrementare o decrementare X o Y, X è incrementato o decrementato di 2 perché si lavora su uno SPRITE multicolore. Se X o Y vanno oltre il bordo della griglia, si fa comparire il cursore sul lato opposto.
190 Se si è battuta la freccia verso sinistra, le corrispondenti 2 posizioni orizzontali sono messe al ?
195 Se si è battuto uno dei tre colori possibili, in R viene messo il corrispettivo ASCII del numero battuto (49 per un 1,50 per un 2, e 51 per un 3) ed il valore di R viene messo nelle due posizioni orizzontali corrispondenti nella matrice A.
200 La gestione del calcolo dello SPRITE viene fatta a partire dalla riga 300
210 Questo comando, come il seguente, è bivalente: Se lo SPRITE è stato raddoppiato su scala X, viene riportato alle dimensioni normali; se è di dimensioni normali viene raddoppiato. In questa riga c'è un esempio significativo di istruzione ABS. Vediamo cosa succede. Se lo SPRITE è stato raddoppiato, in V+29 troveremo un 4 (per lo SPRITE N. 2) e 4-4 fa 0, per cui mettendo 0 in V+29 riporteremo lo SPRITE alle dimensioni normali. Se invece era già di dimensioni normali, con il PEEK (V+29) troveremo 0 e 0-4 fa -4, ma l'istruzione ABS ci dà il valore assoluto di un numero e senza segno, per cui ci dà 4». Facendo il POKE di 4 in V+29 raddoppieremo la dimensione orizzontale dello SPRITE.
220 Stesso discorso fatto alla riga 210, solo che qui è coinvolta la dimensione verticale.
230 Il risultato dello SPRITE in DATA è gestito dalla riga 380.
240 Per il salvataggio o il caricamento da nastro e per un nuovo schermo o per finire si va alla riga 460.
250-280 A seconda del tasto funzione scelto si mette in R un certo valore, dopodiché si va alla linea 400. Da notare che R è il numero di registro del VIC II (in questo caso) e che 33 è per il colore 0 di sfondo, 37 è il registro multicolore 0, 41 è il colore dello SPRITE N. 2 e 38 è il registro multicolore 1.
290 Stampa il cursore. In questo caso R è la locazione attuale del cursore, C è il carattere che c'è sotto il cursore messo in REVERSE (sommandogli 128).
300-320 Calcolo dello SPRITE e cioè dei valori da usare nel POKE per costruire lo SPRITE. Sappiamo che tutti i valori della griglia sono nella matrice A, che contiene un 49 se è stato premuto il colore 1, 50 per il colore 2 e 52 per il colore 3. Si tratta ora di prendere 8 elementi orizzontalidella griglia e trasformarli in un byte,considerando anche il fatto che gli elementi devono essere presi in coppia. Viene letto un elemento, gli viene sottratto 48 in modo tale che il valore sia nuovamente 0, 1, 2, o 3 e viene messo in Q. Poi (linea 320) viene costruito il valore da mettere nella matrice B, utilizzando l'elevazione a potenza perché, come ben sapete, i valori dei bit in un byte sono tutti in potenza di 2.
Questo viene fatto per tutti gli elementi della matrice A, che sono usati per riempire la matrice B con i 63 valori necessari. Subito dopo (sempre linea 320) vengono scritti in memoria (a 831+X) i dati dello SPRITE appena costruito, e si ritorna alla lettura della tastiera.
330-360 Subroutine (chiamata alla linea 460) che ristampa il comando battuto, chiede conferma, mette la risposta in N$ e posiziona il cursore.
370 Subroutine (chiamata alla riga 390) che si rifà alla precedente, ma chiedendo stavolta quando si vuole continuare.
380 Stampa della matrice B, contenente tutti i valori rnecessari per costruire lo SPRITE. La froma «strana» dei comandi di stampa è dovuta al fatto che si vogliono stampare 7 righe di 9 dati ciascuna.
390 Dopo aver stampato la matrice, va a 370 per stampare il messaggio «CONTINUA» e dopo va a 60 per ristampare lo schermo con la matrice.
400-450 Qui troviamo la gestione della colorazione dello SPRITE (gestita con i tasti funzionali). Quando si arriva qui R contiene il numero del registro del VIC II interessato (vedere commento alle righe 250-280). Per prima cosa si legge il contenuto attuale del registro e lo si mette in C, dopo avergli datto un AND con 15 perché il valore netto non può e non deve superare 15. Poi lo si incrementa di 1 (cambiando quindi il codice del colore) ponendo attenzione a che non superi il valore massimo di 15 (in questo caso viene messo a zero). Alla linea **410** viene scritto, nel registro selezionato, il nuovo valore e si comincia a posizionare il cursore sulla prima riga dei comandi (quella col comando di cancellazione). Le linee da 410 a 440 servono giusto a posizionarsi sullo schermo di fianco alla scritta «CANC» se si è cambiato il colore dello schermo oppure di fianco alle scritte 1, 2 e 3 se si è cambiato un colore dello SPRITE. Il nuovo colore viene «scritto» alla linea 450.

Remarks Listato - Sprite Editor

460-480 Qui comincia la gestione dei rimanenti 4 comandi. Se il comando è un «N» (Nuovo schermo) si va alla linea 20 per rifare tutto. Se il comando riguarda il salvataggio o la lettura o la fine si chiede conferma (perché in ogni caso si distruggerà quello che si è fatto finora) facendo un GOSUB a 330. Se non c'è conferma si riprende il giro. Se c'è conferma, si analizza il comando. Se il comando è «F» (fine) si «spegne» lo Sprite e si finisce il programma (linea 480).
490 Se si giunge qui, il comando riguarda il salvataggio o la lettura di uno SPRITE da/su cassetta, per cui si «spegne» lo SPRITE e si chiede il nome, che servirà sia a scriverlo che a leggerlo.
500 Se il comando è «L» (lettura) si fa la OPEN per leggerlo e si va a 520.
510 Se il comando è «S» (scrittura) si scrive tutta la matrice B su nastro e si ricomincia da capo mostrando di nuovo lo SPRITE.
520-550 Continuazione del comando «L» (lettura da nastro). Si comincia col leggere tutti i dati necessari e li si mette nella matrice B. A questo punto bisogna riempire la matrice A, che sappiamo contenere non già i dati di B, ma una loro estrapolazione, in quanto ogni dato (o byte) in B è rappresentato da ben otto dati (o bytes) in A. Ed ecco quindi i calcoli fatti alle linee 530-540, che non fanno altro che convertire un dato in otto dati. Infine si resetta il posizionamento del cursore e si va a mostrare il lavoro fatto.
560-570. Nomi di tutti i colori disponibili, usati per scriverli nella parte «comandi» dello schermo.

appunto a poter indirizzare le posizioni orizzontali da 257 a 320. Questa complessità è necessaria per mantenere una così potente capacità grafica.
Gli Sprite standard (e cioè colorati interamente con un solo colore), possono essere mostrati in ognuno dei sedici colori disponibili. Gli Sprites multicolore invece possono essere colorati con quattro diversi colori per ogni Sprite, ed i colori sono determinati considerando i bit a coppie. Se la coppia è formata da due zeri (00) viene selezionato il colore dello sfondo (cioè non si vede niente), «01» seleziona il colore che è stato posto nel registro multicolore numero 0, «10» il colore che è stato posto nel registro di colore standard per quello Sprite, e «11» il colore che è stato messo nel registro multicolore numero 1. Usando gli Sprites nel modo multicolore, la definizione orizzontale viene un po' a mancare, in quanto avremo dei punti (rispetto agli Sprite monocolore) che sono larghi due volte un punto dello schermo.
Ogni Sprite può essere raddoppiato in larghezza e/o in altezza.
Per gestire «dolcemente» l'entrata e l'uscita degli Sprites sullo schermo, le posizioni possibili per X e Y sono oltre la porzione visibile sullo schermo (512x256 invece che 320x200). In questo modo è possibile vedere «spuntare» uno Sprite lentamente da qualunque parte dello

Listato - Sprite Editor

```
  9 REM INIZIALIZZAZIONE GENERALE
 10 POKE 53281,6:DIM A(21,24),B(63), A$(15): X=0: Y=0: R=0:
    C=0: S=1039: S1=55311
 19 REM INIZIALIZZAZIONE DEL VIC II
 20 V=53248: POKE V+21,0: POKE V+23,0: POKE V+29,0: RESTORE
    :FOR X=0 TO 15: READ A$(X): NEXT
 29 REM RIEMPIMENTO MATRICI A E B
 30 PRINT"(CLR)": FOR R=1 TO 21: FOR C=1 TO 24: A(R,C)=46 :
    NEXT: NEXT: FOR X=1 TO 63: B(X)=0: NEXT
 39 REM POSIZIONAMENTO SPRITE N.2
 40 POKE V+4,60: POKE V+5,200: POKE 2042,13: POKE V+37,0 :
    POKE V+41,14: POKE V+38,1
 50 FOR X=1 TO 63: POKE 831+X,B(X): NEXT: POKE V+21,4: POKE
    V+28,4
 59 REM STAMPA DELLA LISTA DEI COMANDI
 60 PRINT"(CLR) (CG) EDITOR DI SPRITE (CG)"
 70 PRINT"← CANC."
 75 PRINT"1 MC 0 - NERO"
 80 PRINT"2 SC   - BLU CH."
 85 PRINT"3 MC 1 -BIANCO"
 90 PRINT"= CALCOLA SPRITE"
 95 PRINT"X CAMBIA IN 'X'"
100 PRINT"Y CAMBIA IN 'Y'"
105 PRINT"B DATI DEL BASIC"
110 PRINT"N NUOVO SCHERMO"
115 PRINT"S FA' IL SAVE"
120 PRINT"L FA' IL LOAD"
125 PRINT"F FINE"
129 REM DISEGNO DELLA GRIGLIA
130 Y=0: FOR R=1 TO 21: FOR C=1 TO24: Y=Y+1:POKE S+Y,A(R,C)
    : POKE S1+Y,14: NEXT: Y=Y+16: NEXT
140 X=1: Y=1: GOTO 290
149 REM LETTURA COMANDO BATTUTO
150 GET A$: IF A$="" THEN 150
160 P=S+X+(Y-1)*40: C=A(Y,X): POKE P,C: POKE P+1,C
170 IF A$="(CB)" THEN Y=Y+1: IF Y>21 THEN Y=1
175 IF A$="(CA)" THEN Y=Y-1: IF Y<1  THEN Y=21
180 IF A$="(CD)" THEN X=X+2: IF X>24 THEN X=1
185 IF A$="(CS)" THEN X=X-2: IF X<1  THEN X=23
190 IF A$="←" THEN A(Y,X)=46: A(Y,X+1)=46
195 IF A$>"0" AND A$<"4" THEN R=49+VAL(A$): A(Y,X)=R:
    A(Y,X+1)=R
200 IF A$="=" THEN 300
210 IF A$="X" THEN POKE V+29,ABS(PEEK(V+29)-4)
220 IF A$="Y" THEN POKE V+23,ABS(PEEK(V+23)-4)
230 IF A$="B" THEN 380
240 IF A$="L" OR A$="S" OR A$="N" OR A$="F" THEN 460
250 IF A$="(F1)" THEN R=33: GOSUB 400
260 IF A$="(F3)" THEN R=37: GOSUB 400
270 IF A$="(F5)" THEN R=41: GOSUB 400
280 IF A$="(F7)" THEN R=33: GOSUB 400
289 REM POSIZIONAMENTO CURSORE
290 P=S+X+(Y-1)*40: C=(A(Y,X)+128): POKE P,C: POKE P+1,C:
    GOTO 150
```

Listato - Sprite Editor

```
299 REM CALCOLO DELLO SPRITE
300 Y=0: FOR R=1 TO 21: FOR X=0 TO 2: Y=Y+1: B(Y)=0: FOR C=1
    TO 7 STEP 2: Q=A(R,X*8+C)-48
310 IF Q<0 OR Q>3 THEN Q=0
320 B(Y)=B(Y)+2↑(7-C)*Q: NEXT: NEXT: NEXT: FOR X=1 TO 63:
    POKE 831+X,B(X): NEXT: GOTO 140
329 REM STAMPA RICHIESTA DI CONFERMA DEL COMANDO
330 PRINT"(REV)"A$": SI O NO"
340 FOR X=1 TO 10: GET N$: NEXT
350 GET N$: IF N$="" THEN 350
360 PRINT"(CA)       (16 spazi)   (CA)": RETURN
370 PRINT"(REV)CONTINUA": GOTO 340
379 REM STAMPA MATRICE B (DATI DELLO SPRITE)
380 PRINT"(CLR)": FOR X=1 TO 7: PRINT"DATA";: FOR Y=1 TO 9:
    PRINT B((X-1)*9+Y)",";: NEXT
390 PRINT"(CS) ":NEXT: PRINT: GOSUB 370 : GOTO 60
399 REM GESTIONE DELLA COLORAZIONE DELLO SPRITE
400 C=PEEK(V+R) AND 15: C=C+1: IF C>15 THEN C=0
410 POKE V+R,C: PRINT"(HOME). (3 CG)";: IF R=33 THEN 450
420 PRINT"(CG)";:IF R=37 THEN 450
430 PRINT"(CG)";:IF R=41 THEN 450
440 PRINT"(CG)";
450 PRINT"(7 CD)"A$(C)"  ":RETURN
459 REM GESTIONE COMANDI RIMANENTI
460 GOSUB 330: IF N$<>"S" THEN 290
470 GET N$: GET N$: IF A$="N" THEN 20
480 IF A$="F" THEN POKE V+21,0: POKE V+28,0: PRINT"(4 CG)":
    END
489 REM GESTIONE COMANDI 'L' O 'S'
490 PRINT"(CLR)": POKE V+21,0: INPUT"NOME DELLO SPRITE";N$:
    PRINT
500 IF A$="L" THEN OPEN1,1,0,N$: GOTO 520
510 OPEN1,1,1,N$: FOR X=1 TO 63: PRINT#1,B(X): NEXT:CLOSE1:
    GOTO 50
519 REM CONTINUAZIONE COMANDO 'L'
520 FOR X=1 TO 63: INPUT#1,B(X): NEXT: CLOSE1: PRINT"(CG)
    CACOLO LA MATRICE"
530 Y=0: FOR R=1 TO 21:  FOR X=0 TO 2: Y=Y+1: FOR C=2 TO 8
    STEP 2: Q=X*8+C: P=2↑(8-C)
540 S=B(Y) AND (P*3): A(R,Q)=46: A(R,Q-1)=S/P+48
550 NEXT: NEXT: NEXT: S=1039: GOTO 50
559 REM NOMI DEI COLORI, DA CARICARE IN A$(15)
560 DATA NERO., BIANCO, ROSSO, CYAN, PORPORA, VERDE, BLU.,
    GIALLO
570 DATA ARANCIO, MARRONE, ROSSO C, GRIGIO1, GRIGIO2,
    VERDE C, BLU CH., GRIGIO3
```

Lista variabili - Sprite Editor

Lista delle variabili

A (21,24) - Matrice contenente la raffigurazione della griglia mostrata sullo schermo
B (63) - Matrice contenente i dati per costruire lo SPRITE
A$ (15) - Matrice contenente i nomi dei colori disponibili
X,Y - Posizioni orizzontale e verticale del cursore. Usati anche con altri significati.
R - Riga della matrice A usata anche con altri significati (registro VIC II, etc.)
C - Colonna della matrice A usata anche con altri significati.
S - Indirizzo della memoria video per gestire il disegno della griglia
S1 - Indirizzo della memoria colore, usata per gestire il disegno della griglia.
Q - Valore del colore selezionato (da 0 a 3)
V - Indirizzo del VIC II
A$ - Comando battuto

schermo.

Prima ho menzionato le priorità. Gli Sprites, di per sè, hanno delle priorità fisse uno rispetto l'altro: lo Sprite 0 ha una priorità più alta dello Sprite N. 1, l'1 ha una priorità più alta dello Sprite N. 2 e così via. Però ogni Sprite ha la priorità nei confronti dei dati sullo sfondo (testo o sfondo vuoto) che può essere scelta a piacere. Da notare che gli oggetti in priorità più alta «passano sopra» a quelli in priorità più bassa, senza «cancellarli».

Gli scontri sono avvertiti dal VIC II, e i bits appropriati sono messi a 1 in due registri. I bit messi a uno in questa maniera, saranno ripristinati al valore iniziale di zero appena il vostro programma li avrà letti. In questo caso l'intero registro viene azzerato.

C'è un registro per la collisione fra due Sprites, ed uno per la collisione tra Sprite e sfondo.

Alcune limitazioni possono essere ovviate usando tecniche di programmazione più sofisticate. Per esempio è possibile avere più di otto Sprites sul video mostrati contemporaneamente, usando una tecnica detta «raster interrupt». Ancora, potete avere (visto l'abbondanza di memoria) molti Sprites definiti in memoria, e potete cambiare i puntatori a seconda di quale vi serve.

Creazione di uno Sprite

Per creare uno Sprite, dovete disegnarlo in una griglia di 24x21 punti. Dopodiché dovete convertire l'insieme dei punti in una riga in tre bytes di dati, usando il codice binario. Per ogni bytes si sommano i valori dei singoli bits messi a uno. Tutto questo è spiegato, abbastanza chiaramente, nel manuale del 64. Da tutto ciò si rileva che creare gli Sprites non è eccessivamente difficile, una volta presa la mano, ma terribilmente tedioso. Il programma oggetto di questo articolo è appunto un «Edi-

tor» di Sprites e cioè un «attrezzo» per costruire velocemente gli Sprites. I comandi che accetta sono tutti attuati con un tasto: potete visualizzare lo Sprite che state costruendo, fare delle modifiche se volete, farvi mostrare i dati da usare poi nel vostro programma, scriverlo su nastro od eventualmente rileggerlo per fargli ulteriori modifiche. E scusate se è poco.
Una volta fatto partire il programma, tutti i comandi disponibili sono stampati sulla parte sinistra dello schermo. Sulla parte destra invece ci sarà una griglia di 24x21 punti, che è usata per creare/modificare lo Sprite. Per muovere il cursore (che sarà posizionato inizialmente in alto a sinistra) usate i soliti tasti di controllo del cursore. Se volete «accendere» un punto nello Sprite, premete i tasti 1, 2 o 3. A questi tasti sono associati tre colori (inizialmente nero, azzurro e bianco) che potete cambiare premendo i tasti funzionali 3, 5, e 7 rispettivamente.
Mano a mano che cambiate i colori vedrete che anche le descrizioni dei tasti 1, 2 e 3 cambieranno. MC di fianco ai tasti 1 e 3 significa Multi-Colore, mentre SC di fianco al tasto 2 significa Singolo Colore: si riferiscono ovviamente ai registri del VIC II. È importante ricordarsi di ciò perché se vorrete gli stessi colori nel programma che userà lo Sprite, dovrete mettere gli stessi colori nei registri corrispondenti (Registri multicolore 0 e 1 e registro di colore dello Sprite).
Se invece volete cambiare il colore di tutto lo schermo potete farlo battendo il tasto funzione **F1**. Se volete cancellare una coppia di punti precedentemente accesa, lo potete fare con il tasto «». In qualunque momento potete vedere lo Sprite come sarà mostrato nelle sue dimensioni reali premendo il tasto «=»: lo Sprite vi comparirà nella parte in basso a sinistra dello schermo. Se cambiate i colori vedrete che anche il «piccolo» Sprite cambierà colore. Ricordatevi comunque di battere il tasto di calcolo dello Sprite (l'uguale) **prima** di visualizzare i dati per la sua futura costruzione (con il tasto «B») o prima di salvarlo su nastro (con il tasto «S»). Questo perché solo quando voi battete «=» il programma si calcola tutte le informazioni per creare lo Sprite, e **non** mentre voi lo disegnate nella griglia.
Anche se fate delle modifiche al disegno, queste non saranno riportate sul «piccolo» fino a che non batterete l'«=». Le altre cose che potete fare sono l'espansione orizzontale del «piccolo» Sprite, come pure quella verticale (battendo rispettivamente X e/o Y). Per ricominciare a disegnare un altro Sprite battete il tasto «N» (nuovo schermo) e azzererete tutti i dati. Battendo invece «L» caricherete uno Sprite già fatto da nastro mentre il tasto «F» serve a far finire il programma.

Note sul listato

Nel listato del programma **CG** significa «Cursore Giù»; **CA** invece vuol dire Cursore in Alto, **CD** cursore a destra e **CS** cursore a sinistra **F1, F3, F5, F7** sono i rispettivi tasti funzionali.
«**CLR**» vuol dire Clear Screen, **REV** vuol dire REVERSE ON. Qualche volta davanti ai tasti di movimento cursore c'è un numero: vuol dire di battere quel tasto un certo numero di volte. Per esempio: (3 CG) vuol dire di battere tre volte il tasto di Cursore. Un'ultima cosa: le ultime due righe del programma contengono i nomi dei colori, e dopo alcuni nomi c'è un punto: battetelo (non è un errore di stampa). Serve a fare in modo che quando cambiate colore non resti, sotto il nuovo, qualche rimasuglio del nome del colore che c'era prima. Per la spiegazione dettagliata del programma vi rimando alle REMarks. Auguri.

Roll up e roll down per Vic 20

Questo piccolo programma vi permetterà di listare i vostri programmi ed eventualmente di ritornare su una linea appena listata.

L'editing da schermo essendo di soli 22 caratteri richiede, se il programma è abbastanza lungo, parecchio tempo e molta noia mantenendo la possibilità di fare altri errori. Un buon debug si fa o con listato su carta, o avendo a disposizione la possibilità di listare in modo bidirezionale: bene con questo programmino anche voi che possedete il Vic 20 potrete farlo.

La linea **1001** determina l'indirizzo della locazione di memoria di partenza (GO) per ogni particella di memoria del VIC. La linea **1002** calcola il numero delle linee (LN) del programma da listare. La linea **1003** prepara il video per il listato della linea, dopodiché continua il programma. Appena terminata la lista, il programma termina, e questo è gestito dalla linea **1004** che controlla la lista e continua in sequenza dato il comando goto 1010. In linea **1010** e **1030** sono le linee che vi danno la possibilità di rollappare o rolldaunare (belli no!), premendo o il segno del minore (<) o il segno del maggiore (>) per andare rispettivamente in su o in giù.
Attenzione alla linea **1100** che controlla il prossimo O in Basic, ossia la prossima fine linea basic e che quindi vi rimanda a calcolare il prossimo numero di linea. La linea **1200** controlla in modo routine la fine della linea precedente per potervi eventualmente ritornare. Per poter usare bene questo mini programma dovrete quindi eliminare gli O negli indirizzi che determinano il numero di linea, non amettendovi uno O in questa locazione di memoria.
Onde evitare di ribattere sempre questo seppur corto programma al momento di una LOAD ad un programma applicativo, eseguite:
CLR: PRINT PEEK (45): PRINT PEEK (46)
Battete quindi la linea immediatamente seguente
POKE 43, PEEK (45)-2: POKE 44, PEEK (46)
Tale linea vi permette di spostare l'inizio del Basic a due byte prima della fine del

Listato - Roll up

```
1000 REM *** SUGIULST ***
1001 GO=PEEK(44)*256+PEEK(43)-1
1002 LN=PEEK(GO+3)+PEEK(GO+4)*256
1003 PRINT " GOTO 1010 ":PRINT
     "LIST":LN
1004 POKE631,19:POKE632,17
1005 POKE633,31:POKE634,13
1006 POKE695 19:POKE636,13
1007 POKE198,6:END
1010 IF PEEK(197)=60 THEN 1100
1020 IF PEEK(197)=62 THEN 1200
1030 GOTO 1010
1100 IF PEEK(GO+5)<>0 THEN GO=GO+1
1105 GOTO 1100
1110 GO=GO+5 :GOTO 1002
1200 GO=GO-1:IF PEEK(GO)=0 AND
     PEEK(GO-4)<>0 AND
     PEEK(GO-3)<>0 THEN 1002
1210 GOTO 1200
```

programma (sarà necessario un nullo o uno 0 per iniziare a caricare un nuovo programma).
Ora caricate il vostro programma che chiamerete «SUGIULST» (o se no chiamatelo un po' come vi pare), resettate i puntatori basic con : POKE 42,1; POKE 44,16 per il vostro VIC inespanso. Dopodiché cominciate ad editare battendo RUN 1000.
Avrete la possibilità di verificare linea per linea il vostro programma senza doverlo listare su carta dopo ogni modifica.

Il «Simon» sul C64

Vi proponiamo un programma inviato da **Roberto Anglesio** di Torino: una interessante realizzazione su C64 del famosissimo gioco «Simon».
Il gioco originale è costituito da un apparecchio circolare capace di emettere delle sequenze di suoni in tonalità differenti. Una volta ascoltata la sequenza il giocatore deve riprodurla esattamente, agendo su uno dei quattro settori circolari, colorati differentemente e posti sulla parte alta dell'apparecchio.
All'inizio «Simon» emette una nota sola ma, a mano a mano che il giocatore riesce a riprodurre i vari suoni senza sbagliare, il numero di note emesse si incrementa di una.
Tutto questo è stato riprodotto sul Commodore ed il gioco può dirsi identico all'originale per quanto concerne la sua meccanica.
I quattro settori circolari di Simon sono sostituiti dai quattro tasti funzione del C64; inoltre, poichè detti settori (nel gioco originale) si illuminano per facilitare il compito del giocatore, lo schermo del computer si colora differentemente a seconda della nota emessa e, al centro dello schermo stesso, appare un numero che corrisponde al tasto funzione interessato.
Quando il giocatore sbaglia, il computer emette un suono che lo avverte dell'errore e gli indica poi il numero di note che è riuscito a mettere in sequenza.
Attenzione però: il giocatore non può pensare troppo a lungo perché dopo un certo

Convenzione usata per la scrittura dei caratteri grafici - Simon

(cls)	=	CLR HOME + SHIFT
(n spazi)	=	Numero di spazi
()	=	1 spazio
(176.96 ecc)	=	Rappresenta il codice ASCII dei caratteri, per visualizzare il carattere basta impostare: ?CHRS(n.)160 corrisponde allo spazio
(home)	=	CLR HOME
(Cs)	=	Cursore a sinistra

Solo alle linee 90, 100, 120, 140, 280 e 290 ho disegnato direttamente il carattere (Q+Shift e W+Shift)

Variabili usate nel programma Simon

A (4,4)	=	Tabella corrispondente alla tastiera
G	=	Indica a quale giocatore va riferita la mossa
G1$,G2$	=	Nomi dei giocatori
M$	=	Lettera corrispondente alla mossa
M	=	Codice ASCII relativo a M$
S	=	Corrisponde al valore (pedina) da inserire nella tabella A
L	=	Riga o colonna corrispondente alla lettera di input
C,D,E,F	=	Valori usati per aggiornare lo schermo
H	=	Indica il tipo di verifica da applicare alla somma
A1	=	Somma parziale
T	=	Somma
J$	=	Nome del vincitore
A,B	=	Coordinate della scacchiera
P	=	ASCII delle lettere da scrivere durante l'inizializzazione dello schermo

Note a «Simon»

Nelle istruzioni di stampa o di definizione di stringhe le diciture tra parentesi indicano rispettivamente:
- **CLR/HOME** = tasti SHIFT+CLR/HOME
- **CRSR G** = cursore giù
- **CRSR D** = cursore a destra
- **CTRL n.** = tasti CONTROL+n. dove n. è il tasto numerico specificato
- **C=x** = tasti SIMBOLO COMMODORE+x dove x è il tasto specificato
- **HOME** = tasto CLR/HOME da solo
- **F1, F3, F5, F7** = tasti di funzione

Se le combinazioni di tasti sopra descritte devono essere digitate più di una volta, è indicato subito dopo l'apertura della parentesi il numero di volte relativo.
Nella riga 130 i caratteri (C) devono essere digitati così come sono.

Descrizione del programma «Simon»

Righe 10-20 viene preparata la matrice contenente le diverse combinazioni di note; tali note vengono determinate casualmente
Righe 30-120 costituzione stringhe per il titolo animato
Righe 130-160 intestazione e animazione titolo
Riga 165 viene azzerato il buffer di tastiera
Riga 170 aspetta che un tasto venga premuto per incominciare
Righe 175-176 informa sul modo di giocare
Righe 180-190 pulisce lo schermo e setta il SID
Righe 200-220 esegue la sequenza di suoni
Riga 225 azzera il buffer di tastiera
Righe 230-240 richiede al giocatore la sequenza di suoni
Righe 250-280 assegna un valore a seconda del tasto premuto
Riga 290 se il tasto premuto non è uno dei quattro ammessi torna a 240
Riga 300 se il tasto premuto corrisponde ad un suono sbagliato, va alla routine di errore
Righe 310-320 esegue la sequenza impostata dal giocatore
Righe 1000-1300 a seconda del suono da eseguire assegna diversi valori per il colore dello schermo, la frequenza della nota, il numero corrispondente al tasto di funzione da premere e il colore di tale numero
Riga 1400 emette un suono che segnala l'errore
Riga 1410 stampa il messaggio relativo al numero di note che si sono riuscite a mettere in sequenza
Riga 1420 pulisce lo schermo, punta ad una nuova sequenza di note e torna all'inizio del gioco
Righe 1500-1520 routine che emette il suono, colora lo schermo, stampa il numero relativo al tasto di funzione da premere.

Listato Simon

```
5 REM + SIMON PER COMMODORE 64 +
6 REM + ROBERTO ANGLESIO 011/616128 +
10 PRINT"(CLR/HOME)(2 CRSR G)(9 CRSR D)PLEASE WAIT....":DIMSE(50,30)
20 FORK=1TO50:FORJ=1TO30:SE(K,J)=INT(RND(0)*4)+1:NEXTJ:NEXTK
30 F$(1)="(CTRL 8)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)
   (C=B)(CTRL 0)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
40 B$(1)="(C=B)(5 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CTRL 9)(2 C=B)(CTRL 0)
   (2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CRSR D)(C=B)"
50 C$(1)="(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)(C=B)
   (CTRL 0)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CRSR D)(2 C=B)"
60 D$(1)="(3 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)
   (C=B)(CTRL 0)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
70 E$(1)="(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)(C=B)
   (CTRL 0)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
80 F$(2)="(CTRL 6)(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CTRL 0)(C=B)
   (CTRL 9)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
90 B$(2)="(CTRL 9)(C=B)(5 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CTRL 0)(2 C=B)
   (CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CRSR D)(C=B)"
100 C$(2)="(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 0)
   (C=B)(CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CRSR D)(2 C=B)"
110 D$(2)="(CTRL 9)(3 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CTRL 0)(C=B)
   (CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
120 E$(2)="(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 0)
   (C=B)(CTRL 9)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
130 POKE53280,6:PRINT"(CLR/HOME)(17 CRSR G)(10 CRSR D)(C) DER BESTE 1984":
   I=2:D=1
140 PRINT"(6 CRSR G)(13 CRSR D)(CTRL 9)(CTRL 4) HIT ANY KEY (C=7)(CTRL 0)";
150 PRINT"(HOME)(4 CRSR G)":D=-D:I=I+D
160 PRINTSPC(6)F$(I):PRINTSPC(6)B$(I):PRINTSPC(6)C$(I):PRINTSPC(6)D$(I):
   PRINTSPC(6)E$(I)
165 POKE 198,0
170 FORT=1TO30:GETA$:IFA$=""THENNEXT:GOTO150
175PRINT"(CLR/HOME)(2 CRSR G)(5 CRSR D)PER SUONARE USARE I TASTI DI"
176 PRINT"(CRSR G)(5 CRSR D)FUNZIONE 'F1' 'F3' 'F5' 'F7'":FORT=1TO2000:NEXTT
180 PRINT"(CLR/HOME)":SI=54272:FH=SI+1:W=SI+4:A=SI+5:H=SI+6:L=SI+24:K=1:J=1
190 POKEL,15:POKEA,0:POKEH,160
200FORP=1TOJ:RO=1
210 ONSE(K,P)GOSUB1000,1100,1200,1300
220 NEXTP
225 POKE198,0
230 FORP=1TOJ
240 FORR=1TO500:GETA$:IFA$=""THENNEXTR:GOTO1400
250 IFA$="(F1)"THENN=1:GOTO300
260 IFA$="(F3)"THENN=2:GOTO300
270 IFA$="(F5)"THENN=3:GOTO300
280 IFA$="(F7)"THENN=4:GOTO300
290 NEXTR
300 IFSE(K,P)<>NTHEN1400
310ONNGOSUB1000,1100,1200,1300
320 NEXTP:FORT=1TO200:NEXTT:J=J+1:GOTO200
1000 SC=7:SU=30:NR=49:SF=0:GOTO1500
1100 SC=5:SU=40:NR=51:SF=0:GOTO1500
1200 SC=2:SU=50:NR=53:SF=1:GOTO1500
```

Listato Simon

```
1300 SC=6:SU=60:NR=55:SF=1:GOTO1500
1400 POKEFH,10:POKESI,50:POKEW,33:FORT=1TO200:NEXTT:POKEW,0
1410 PRINT"(CLR/HOME)(8 CRSR G)     HAI MESSO IN SEQUENZA"J-1"NOTE!":
     FORT=1TO1500:NEXTT
1420 PRINT"(CLR/HOME)":K=K+1:J=1:GOTO200
1500 POKE55795,SF:POKE1523,NR
1505 POKEFH,SU:POKESI,100:POKEW,17:POKE53281,SC:POKE53280,SC:FORT=1TO200:
     NEXTT:POKEW,0
1510 IFRO=1THENFORT=1TO200:NEXTT:RO=0
1520 POKE53281,0:POKE53280,0:POKE1523,32:RETURN
```

intervallo di tempo, se nessuno dei tasti funzione è stato premuto, il computer si comporta come se si fosse commesso un errore e si deve ricominciare.

«Black Box» sul VIC 20

Vi proponiamo una bella soluzione di «Black Box», gioco di riflessione alla scoperta delle leggi della natura, proposto su Personal Computer n. 2, nella versione per **Vic 20** dell'amico **Fulvio Forti** di Novate Milanese. Più che di una soluzione si tratta in realtà di uno studio con tanto di bibliografia (in lingua inglese: E.W. Solomon: Black Box Game Book. Strategy Games Publ., London, 1977), in cui sono riportate tra l'altro le regole originali che noi avevamo lievemente semplificato nel caso di deflessioni sul bordo della scatola nera. Bene, note le leggi di riflessione, deflessione e assorbimento dei raggi che – lanciati all'interno della scatola nera – servono a darvi un'idea della disposizione degli atomi in essa, si tratta soltanto di fare un numero sufficiente di prove per confermare le proprie congetture. Nelle due figure allegate sono riportati esempi di queste leggi nonché la numerazione delle caselle di ingresso dei raggi sulla scacchiera 8x8 su cui è stato programmato il gioco. Per comunicare al Personal

Listato Black Box

```
10 PRINT"♥QQQQQQ"TAB(5)"SCATOLA NERA"
20 PRINT"QQQ"TAB(9)"DI"
30 PRINT"QQQ"TAB(5)"FULVIO FORTI"
40 FORK=1TO5000:NEXTK
50 N=0:HV=7746:C=38466:S$="S]]]]]]]]]]]]]]]QQQ"
60 GOSUB400:GOSUB500
70 GOSUB700:INPUT"]COSA GIOCHI]";E$
80 IFE$="I"THENGOSUB800:GOTO70
90 IFE$="C"THEN190
100 E=VAL(E$):IFE<1ORE>32THEN70
110 N=N+1:IFN>15THENGOSUB700:PRINT"]]TROPPI TENTATIVI
!":FORK=1TO2000: NEXTK:GOTO320
120 GOSUB1000
130 IFU=0THENU$="ASS":FORK=1TO800:NEXTK:GOTO160
140 IFU=ETHENU$="RIF":FORK=1TO800:NEXTK:GOTO160
150 U$=STR$(U):U$="   "+U$:U$=RIGHT$(U$,3)
160 E$="   "+E$:E$=RIGHT$(E$,3)
170 S$=S$+"Q":PRINTS$;
180 PRINTE$"]]"U$:GOTO70
190 FORJ=1TO4
200 K(J)=HV+S(J)-INT(S(J)/10)*10+INT(S(J)/10)*22
210 NEXTJ
220 FORJ=1TO4
230 IFPEEK(K(J))=42THEN250
240 J=4:GOSUB700:PRINTTAB(5)"HAI PERSO !":GOTO270
250 NEXTJ
260 GOSUB700:PRINT"HAI VINTO IN"N"TIRI"
270 FORJ=1TO4
280 POKEK(J),42
290 POKE30720+K(J),2
300 NEXTJ
310 FORJ=1TO800:NEXTJ
320 GOSUB700:PRINT"VUOI RIPROVARE (S/N)?"
330 GETA$:IFA$=""THEN330
340 IFA$="S"THENRUN50
```

la direzione dei propri tentativi è sufficiente infatti indicare un numero compreso tra 1 e 32, essendo da ciò implicito anche il verso prescelto.
Per inserire un asterisco ad indicazione di un atomo della molecola (soltanto al fine di aiutare le proprie considerazioni) bisognerà digitare **I**: con lo stesso comando si potrà togliere un atomo mal piazzato. Con C infine si chiede al Personal di concludere il gioco, controllando se tutti gli atomi piazzati sono al posto giusto (condizione di vittoria) oppure no.
È appena il caso di dire che il gioco si svolge in solitario contro il Personal che nasconde gli atomi (righe 400-470 del listato), che il programma gira sul VIC 20 in configurazione base e che utilizza un algoritmo di numerazione delle caselle lievemente differente da quello descritto, che però è trasparente al giocatore, in quanto è «tradotto» prima di essere presentato a video. I simboli grafici riportati nel listato sono specificati accanto ad esso.

Strategia per scacchiera con «Othello» sul VIC 20

Di origine – si dice – giapponese, Othello è uno dei più interessanti giochi di strategia per scacchiera di recente invenzione. Noto anche come «Reversi», come tutti i grandi giochi ha il fascino di regole elementari, al punto che la sua pubblicità (in Italia è distribuito dalla Clem Toys) parla di «un minuto per impararlo, una vita per diventare campioni».
Su una normale scacchiera 8x8 si inizia a giocare disponendo al centro quattro pedine in quadrato, due del colore di un giocatore e due del

Listato - Black Box

```
35Ø IFA$="N"THENPRINT"♡":END
36Ø GOTO33Ø
4ØØ FORJ=1TO4
41Ø X(J)=INT(RND(1)*8+1)*1Ø+INT(RND(1)*8+1)
42Ø FORK=1TO4
43Ø IFX(J)=S(K)THENK=4:GOTO41Ø
44Ø NEXTK
45Ø S(J)=X(J)
46Ø NEXTJ
47Ø RETURN
5ØØ PRINT"♡"TAB(2)"** SCATOLA NERA **"
51Ø PRINT"Q 33322222"
52Ø PRINT" 21Ø98765"
53Ø PRINT"1........24"
54Ø PRINT"2........23"
55Ø PRINT"3........22"
56Ø PRINT"4........21"
57Ø PRINT"5........2Ø"
58Ø PRINT"6........19"
59Ø PRINT"7........18"
6ØØ PRINT"8........17"
61Ø PRINT" 91111111"
62Ø PRINT"  Ø123456"
63Ø PRINT"SQQ"TAB(13)"ENT  USC"
64Ø PRINTTAB(11)"┌────┬────┐",
65Ø FORK=1TO15
66Ø PRINTTAB(11)"|    |    |",
67Ø NEXTK
68Ø PRINTTAB(11)"└────┴────┘",
69Ø RETURN
7ØØ PRINT"SQQQQQQQQQQQQQQQQQQQQQQ";
71Ø FORK=ØTO43
72Ø POKE8142+K,32
73Ø NEXTK
74Ø RETURN
8ØØ GOSUB7ØØ:INPUT"COORDINATA ORIZ. ";H$
81Ø H=VAL(H$)
82Ø IFH>8ANDH<17THENH=H-8:GOTO85Ø
83Ø IFH>24ANDH<33THENH=33-H:GOTO85Ø
84Ø GOTO8ØØ
85Ø GOSUB7ØØ:INPUT"COORDINATA VERT. ";V$
86Ø V=VAL(V$)
87Ø IFV>ØANDV<9THEN9ØØ
88Ø IFV>16ANDV<25THENV=25-V:GOTO9ØØ
89Ø GOTO85Ø
9ØØ GOSUB7ØØ:INPUT"   SIMBOLO (*,.)";R$
91Ø R=ASC(R$)
92Ø IFR=42ORR=46THEN94Ø
```

Listato - Block Box

```
930 GOTO900
940 POKEHV+H+V*22,R
950 POKEC+H+V*22,6
960 RETURN
1000 IFE<9THENP=E*10:A=1:B=10:GOTO1040
1010 IFE>8ANDE<17THENP=E+82:A=-10:B=1:GOTO1040
1020 IFE>16ANDE<25THENP=(26-E)*10-1:A=-1:B=10:GOTO1040
1030 P=33-E:A=10:B=1
1040 X=A:Y=B
1050 FORJ=1TO4
1060 IFP+X=S(J)THENU=0:RETURN
1070 NEXTJ
1080 FORJ=1TO4
1090 IFP+X+Y<>S(J)THEN1120
1100 J=4:GOSUB1240:IFF=1THENRETURN
1110 GOTO1190
1120 NEXTJ
1130 FORJ=1TO4
1140 IFP+X-Y<>S(J)THEN1170
1150 J=4:GOSUB1240:IFF=1THENRETURN
1160 A=Y:B=ABS(X):GOTO1040
1170 NEXTJ
1180 GOTO1220
1190 FORJ=1TO4
1200 IFP+X-Y=S(J)THENA=-X:GOTO1040
1210 A=-Y:B=ABS(X):GOTO1040
1220 P=P+X:GOSUB1240:IFF=1THENRETURN
1230 GOTO1040
1240 IFP/10-INT(P/10)=0THENF=1:U=P/10:RETURN
1250 IFP>90THENF=1:U=P-82:RETURN
1260 IF(P+1)/10-INT((P+1)/10)=0THENF=1:U=26-5P+1)/10:RETURN
1270 IFP<9THENF=1:U=33-P:RETURN
1280 F=0:RETURN
```

Simboli grafici utilizzati nel listato Black Box

```
♥   SHIFT + CLR/HOME
Q   CRSR in basso
]   CRSR a destra
S   HOME
```

suo avversario. La caratteristica di queste pedine è di essere bifronti, cioè colorate su due lati con i colori dei due contendenti (come certi biscotti che si mangiavano da bambini...). La «mossa» consiste nel deporre una pedina del proprio colore su una casella libera, a patto che «prenda in mezzo» almeno una pedina avversaria, che viene immediatamente girata e cambia dunque colore. Se non è possibile una mossa di questo tipo si è costretti a «passare». Per controllare se una mossa fa cambiare colore a qualche pedina avversaria, bisogna esplorare tutte le direzioni (anche diagonali!) a partire dalla posizione giocata, e vedere se c'è qualche pedina (o gruppo di pedine) avversaria che risulti presa in mezzo.

È chiaro perciò che i quattro angoli della scacchiera sono posizioni di tutta sicurezza (e di grande interesse strategico), perché è impossibile far cambiare ad essi colore. Il gioco prosegue in maniera imprevedibile, perché ad ogni mossa cambiano colore molte pedine (TUTTE quelle che soddisfano le regole) e tra un colpo di scena e l'altro si arriva a riempire tutta la scacchiera, oppure ad una posizione in cui nessuno dei due avversari dispone di mosse valide. Il risultato finale, allora, è dato dal numero di pedine del colore dei due contendenti: si va da uno schiacciante 64-0 (63-1, e così via) ad un risicato 33-31 o addirittura alla parità 32-32.

Non pretendiamo certo qui di esaurire la teoria di Othello, che tuttora vede migliaia di giocatori impegnati in tutta Italia ad allenarsi per il campionato italiano che apre le porte ai mondiali (dominati dai giapponesi, ma con brillanti piazzamenti dei nostri rappresentanti): vi suggeriamo soltanto, se dopo un apprendistato su questo gioco vi interessa l'attività agonistica, di mettervi in contatto con la Clem Toys a Recanati (AN).

Sul listato dell'infaticabile **Vincenzo Leto** di Pisa c'è poco da dire: si gioca in due avversari umani (interessante sarebbe pure giocare contro il computer, ma è un affar serio da programmare), scegliendo il colore delle proprie pedine. Le mosse vanno comunicate alla macchina specificando le coordinate della casella «di arrivo» della pedina. Il vantaggio di giocare a video anziché su una scacchiera sta tutto nel fatto che i controlli sulla validità della mossa, ma soprattutto le pedine avversarie da «girare» sono appannaggio del Personal, laddove molti principianti commettono errori che falsano la partita.

Alla fine, la vittoria di uno dei due giocatori è celebrata visivamente e acusticamente per mezzo della «Bourrée» di J.S. Bach. (L.F.)

Listato Othello

```
1 POKE36879,110:PRINT"OTHELLO"
10 PRINT"DI V.LETO"
15 POKE36878,15:FORI=1TO27:READA%:POKE36875,A%:FORJ=1TO250:NEXTJ,I:S$="
"
20 DATA237,239,225,225,239,237,217,217,237,239,207,212,217,237,235,232,207,228
25 DATA225,223,195,201,207,201,195,223,219
30 DIMC$(7),SC%(9,9)
35 FORI=0TO9:FORJ=0TO9:SC%(I,J)=-1:NEXTJ,I:U=38512
40 C$(1)="":C$(2)="":C$(3)="":C$(4)="":C$(5)="":C$(6)="":C$(7)="":D$="
"
45 FORI=15TO0STEP-1:POKE36878,I:NEXT:POKE36879,28
50 INPUT"1"NOME";A$:GOSUB520
55 IFLEN(A$)>10THENA$=LEFT$(A$,10)
60 GOSUB560:C=H%:IFC<1ORC>7THEN60
65 A$=C$(C)+" "+A$:C$(C)="":INPUT"2"NOME";B$:GOSUB520
70 IFLEN(B$)>10THENA$=LEFT$(B$,10)
75 GOSUB560:D=H%:IFD<1ORD>7ORD=CTHEN75
80 B$=C$(D)+" "+B$:IFC=1THENC=0
85 IFD=1THEND=0
90 Q$=" ":R$=" ":PRINT"
"
95 FORI=1TO7:PRINTR$:PRINTQ$:NEXTI:PRINTR$:PRINT" "
100 PRINT"1 2 3 4 5 6 7 8":PRINT"12345678
"
105 POKE36875,0:POKE36878,15
110 FORI=4TO5:SC%(I,I)=D:SC%(I,9-I)=C:NEXTI
115 FORX=6TO20STEP2:FORY=3TO17STEP2:POKE38400+Y+22*X,1:POKE7680+Y+22*X,160:NEXTY
,X
120 POKE38675,C:POKE38717,C:POKE38673,D:POKE38719,D:POKE8165,30:POKE7787,31
125 FORH1=1TO60
130 FORJ=7710TO7723:POKEJ,32:NEXTJ:PRINT"GIOCA "A$
135 GETW$:GOSUB530:IFW$=""THEN135
140 W=VAL(W$):IFW<1ORW>8THEN135
145 D1=7813+(W-1)*44:POKED1,PEEK(D1)+128
150 GETZ$:GOSUB540:IFZ$=""THEN150
155 Z=VAL(Z$):IFZ<1ORZ>8THEN150
160 D2=7771+(Z-1)*2:POKED2,PEEK(D2)+128
165 IFSC%(W,Z)<>-1THENGOTO550
175 FORL=W-1TOW+1:FORK=Z-1TOZ+1:IFL=WANDK=ZTHEN185
180 IFSC%(L,K)<>-1THEN190
185 NEXTK,L:GOTO550
190 SC%(W,Z)=C:POKEU+2*Z+44*W-23,C:E=-1:G=1:GOSUB710:FORI=W-1TOW+1:FORJ=Z-1TOZ+1
195 IFI=WANDJ=ZTHENE=1:GOTO210
200 IFSC%(I,J)=DTHENGOSUB600
205 G=G+1
210 NEXTJ,I:P$=A$:A$=B$:B$=P$:G=C:C=D:D=G:GOSUB555:NEXTH1:RESTORE
215 K=0:FORI=1TO8:FORJ=1TO8:IFSC%(I,J)=CTHENK=K+1
220 NEXTJ,I:FORF=7705TO7722:POKEF,32:NEXT:PRINT""A$" ==>"K""B$" ==>
"64-K
225 IFK=32THEN245
230 IFK<32THENG%=22:C=D
235 E=1:FORI=1TO27:FORJ=38425TO38442:POKEJ+G%,C:NEXTJ
240 READA%:POKE36875,A%:F=C:C=E:E=F:FORN=1TO100:NEXTN,I:POKE36875,0:GOTO250
245 PRINT" *** PARTITA PATTA **";
250 PRINT" CONTINUATE (S/N)?";
255 GETL$:IFL$=""THEN255
260 IFL$="S"THEN35
265 IFL$="N"THENPRINT"CIAO!":POKE36879,110:POKE36878,0:END
270 GOTO255
510 POKE36878,15:POKE36874,128+INT(RND(1)*127):RETURN
520 PRINT"":FORI=1TO7:PRINTTAB((I-1)*3+2)C$(I)D$""I"":NEXTI:RETU
RN
530 POKE38885,2:FORL=1TO200:NEXT:POKE38885,1:FORL=1TO200:NEXT:RETURN
540 POKE38507,2:FORL=1TO200:NEXT:POKE38507,1:FORL=1TO200:NEXT:RETURN
550 PRINTS$" MOSSA ERRATA ";:GOSUB720:FORL=1TO1000:NEXT:GOSUB555:PRINTS$"
```

Listato Othello

```
"; :GOTO135
555 POKED1,PEEK(D1)-128:POKED2,PEEK(D2)-128:FORF=1TO120:NEXT:RETURN
560 PRINT"[illegible]QUALE COLORE VUOI?"
565 GETH$:IFH$=""THEN565
570 H%=ASC(H$)-48:RETURN
600 ONGOTO610,630,650,670,670,650,630,610
610 A=I+E:B=J+E
615 IFSC%(A,B)=DTHENA=A+E:B=B+E:GOTO615
620 IFSC%(A,B)=-1THENRETURN
625 FORK=ITOA-ESTEPE:SC%(K,K-I+J)=C:POKEU+(2*(K-I+J)-1)+22*(2*K-1),C:GOSUB700:NE
XT:RETURN
630 A=I+E
635 IFSC%(A,J)=DTHENA=A+E:GOTO635
640 IFSC%(A,J)=-1THENRETURN
645 FORK=ITOA-ESTEPE:SC%(K,J)=C:POKEU+2*J+44*K-23,C:GOSUB700:NEXT:RETURN
650 A=I+E:B=J-E
655 IFSC%(A,B)=DTHENA=A+E:B=B-E:GOTO655
660 IFSC%(A,B)=-1THENRETURN
665 FORK=ITOA-ESTEPE:SC%(K,J+I-K)=C:POKEU+2*(I+J)+42*K-23,C:GOSUB700:NEXT:RETURN
670 B=J+E
675 IFSC%(I,B)=DTHENB=B+E:GOTO675
680 IFSC%(I,B)=-1THENRETURN
685 FORK=JTOB-ESTEPE:SC%(I,K)=C:POKEU+2*K+44*I-23,C:GOSUB700:NEXT:RETURN
700 FORF=240TO255:POKE36874,F:NEXT:POKE36874,0:RETURN
710 FORF=240TO255:POKE36875,F:NEXT:POKE36875,0:RETURN
720 FORF=198TO128STEP-1:POKE36876,F:NEXT:POKE36876,0:RETURN
```

Il controllo del cursore

Il nuovo comando che vi proponiamo è una semplice routine per il posizionamento del cursore in qualunque parte dello schermo, senza usare dei trucchi di programmazione BASIC come quello che segue:

100 RS= «(premere il tasto di cursore a destra 39 volte)».
110 D$= «(premere il tasto di cursore in basso 23 volte)».
500 C=22:R=15: REM MUOVE IL CURSORE ALLA COLONNA 22, RIGA 15
510 M$=LEFT$ (R$, C) + LEFT$ (D$, R)
520 PRINTM$.

In questo modo (ovviamente le righe 510 e 520 potrebbero essere chiamate come subroutine, aggiungendo un GOSUB alla linea 500 e un RETURN alla linea 520) si ottiene il posizionamento del cursore sopra descritto, ma non è una brutta tiritera?

Ecco invece una ottima soluzione di cui forniamo il listato in Assembler che voi potete immettere senza problemi. La forma fornita è adatta per un PET con il BASIC 4, con uno schermo piccolo. Le versioni per gli altri PET lavorano esattamente allo stesso modo e qui troverete anche dei DUMPS in esadecimale per i possessori delle differenti versioni.

E vediamo ora i dumps esadecimali per le varie versioni del PET.

Schermo largo, BASIC 4
033A 20 F5 BE 20 D4 C8 E0 18
0342 90 03 4C 00 BF 86 D8 20
034A F5 BE 20 D4 C8 E0 28 B0
0352 F1 86 C6 20 71 E0 60

Schermo piccolo, BASIC 4
033A 20 F5 BE 20 D4 C8 E0 18
0342 90 03 4C 00 BF 86 D8 20
034A F5 BE 20 D4 C8 E0 28 BO
0352 F1 86 C6 20 7F E0 60

80 colonne BASIC 4
033A 20 F5 BE 20 D4 C8 E0 18
0342 90 03 4C 00 BF 86 E0 20
034A F5 BE 20 D4 C8 EO 50 BO
0352 F1 86 E2 20 5F EO 60

Controllo cursore

033A 20 F5 BE	JSR $BEF5;	Controlla se trova una virgola
033D 20 D4 C8	JSR $C8D4;	Prendi un parametro di 1 byte mettilo in x
0340 EO 18	CPX £$18;	Confrontalo con 24 (num. di righe)
0342 90 03	BCC $0347;	Se OK il carry on
0344 4C 00 BF	JMP $BF00;	Se no messaggio «Syntax Error»
0347 86 D8	STX $D8;	Salva il primo parametro in £ D8
0349 20 F5 BE	JSR $BEF5;	Controlla per seconda virgola
034C 20 D4 C8	JSR $C8D4;	Prendi il prossimo parametro
034F EO 28	CPX £$28;	Confrontalo con 40 (num. colonne)
0351 BO F1	BCS $O344;	Se maggiore messaggio «Syntax Error»
0353 86 C6	STX $C6,	Se OK salva il secondo param. in £C6
0335 20 71 E0	JMP $EO71;	Posiziona il cursore
0358 60	RTS	

BASIC 3
033A 20 F8 CD 20 78 D6 E0 18
0342 90 03 4C 03 CE D8 20
034A F8 CD 20 78 D6 E0 28 BO
0352 F1 86 C6 20 5D E2 60

Come si utilizza il tutto? Semplice! I dati che voi avete immesso ora risiedono nel Buffer della seconda unità nastro. Per usare il nuovo comando, in qualunque momento, basta inserire questa linea nel vostro programma:

250 SYS 826, 22, 15
Dove 250 è il numero di riga di programma, 22 è la colonna e 15 è la riga dello schermo dove si vuole mettere il cursore. Quando calcolate la posizione assumete che l'origine (0,0) sia in alto a sinistra sullo schermo. È una buona norma non cercare di mettere il cursore sotto la riga 24 perché ciò potrebbe causare uno scrolling accidentale.

Grafica ad alta risoluzione col VIC 20

di Walter Pistarini

Questo articolo spiega come si può ottenere una grafica di 28160 (ventottomilacentosessanta!) punti sul VIC, con una matrice di 176x160 punti (per i VIC con qualche espansione di memoria, per quelli senza espansione la grafica sarà «solo» di 19584 punti, su una matrice di 136x144 punti).
Il manuale del VIC parla di capacità di grafica ad alta risoluzione. C'è anche una sezione che mostra come fare della grafica a 64x64 punti. Sfortunatamente, non è molto chiaro come gira il tutto. Lo scopo di questo articolo è di gettare un po' di luce nella grafica del VIC.
Per capire la grafica ad alta risoluzione bisogna prima capire come lavorano i caratteri programmabili. Il VIC in realtà non ha un modo di lavorare «grafico», ma ha due possibilità che permettono di creare della grafica. La prima e più importante è la possibilità di cambiare il contenuto del puntatore che normalmente punta alla ROM contenente i caratteri (diciamo meglio, le «informazioni» per costruire i caratteri) e farlo puntare in RAM. L'altra possibilità è che la grandezza di carattere può essere cambiata.

Memorie video e caratteri

Prima di spiegare come queste due «opzioni» si combinino per ottenere una buona grafica, dobbiamo rivedere come i caratteri sono solitamente mostrati sullo schermo. Prima di tutto facciamo un breve ripasso su cosa si intede per «memoria video» e cosa si intende per «memoria caratteri» (character generator in inglese).

Per **«memoria video»** si intende quella parte di memoria del VIC atta a contenere dei «codici» (un codice è lungo un byte) che dicono al VIC quale carattere si vuole vedere in un determinato punto dello schermo. Si potrebbe dire che è la «fotografia» di quello che vedete sullo schermo. C'è una corrispondenza diretta (1 a 1) tra gli indirizzi (o posizioni) nella memoria video e le posizioni sullo schermo. Per fare un esempio: se avete un codice che indica la lettera «A» nella locazione 23 della memoria video, questa lettera «A» vi sarà mostrata nella posizione 23 dello schermo, e cioè come primo carattere della seconda riga, visto che le righe sono lunghe 22 caratteri.

Quindi: Memoria video = foto del video. In un VIC senza espansioni o con l'espansione da 3K, la memoria video comincia all'indirizzo 7680. Se ci sono altre espansioni sarà a 4096 (normalmente).

Per **«memoria caratteri»** si intende quella parte della memoria del VIC (normalmente su ROM) che contiene le informazioni necessarie per «costruire» tutti i caratteri disponibili sul VIC. Più esattamente diciamo che questa memoria dice al VIC, dato un certo codice (preso nella memoria video), quali punti dello schermo (detti PIXELS) bisogna «accendere» o «spegnere» per mostrare il carattere identificato da quel codice.

Quindi: Memoria caratteri = foto dei caratteri disponibili.
Ogni byte della memoria video è usato come un indice per trovare nella memoria caratteri le appropriate «informazioni». Nel mondo di lavorare normale, un carattere è costituito da 8 righe di 8 punti ciascuna. Con la «opzione» che espande la grandezza dei caratteri abbiamo 16 righe di 8 punti ciascuna.
Il primo «codice carattere» che potete selezionare è 0 (una @), e mettendo 0 nella prima locazione di memoria video dite al VIC prima di tutto che volete vedere il carattere con il codice 0 nella prima posizione dello schermo e poi di andare a vedere nella memoria caratteri i primi 8 bytes, che gli diranno quali punti accendere sul video in quella posizione. Questo perché ogni carattere, come già detto, è fatto di 8 righe di 8 punti ciascuna, per cui il VIC avrà bisogno di 8 bytes di 8 bits ciascuno per sapere quali dei 64 punti (8 righe x 8 punti) che formano un carattere deve accendere o spegnere. Ogni byte della memoria caratteri definisce (con i suoi 8 bits) una riga di punti di un carattere. Se il bit è a 1 vuol dire che il corrispondente punto deve essere acceso, se è da 0 il punto dello schermo deve essere spento.
Provate, su un VIC senza espansioni, a battere:

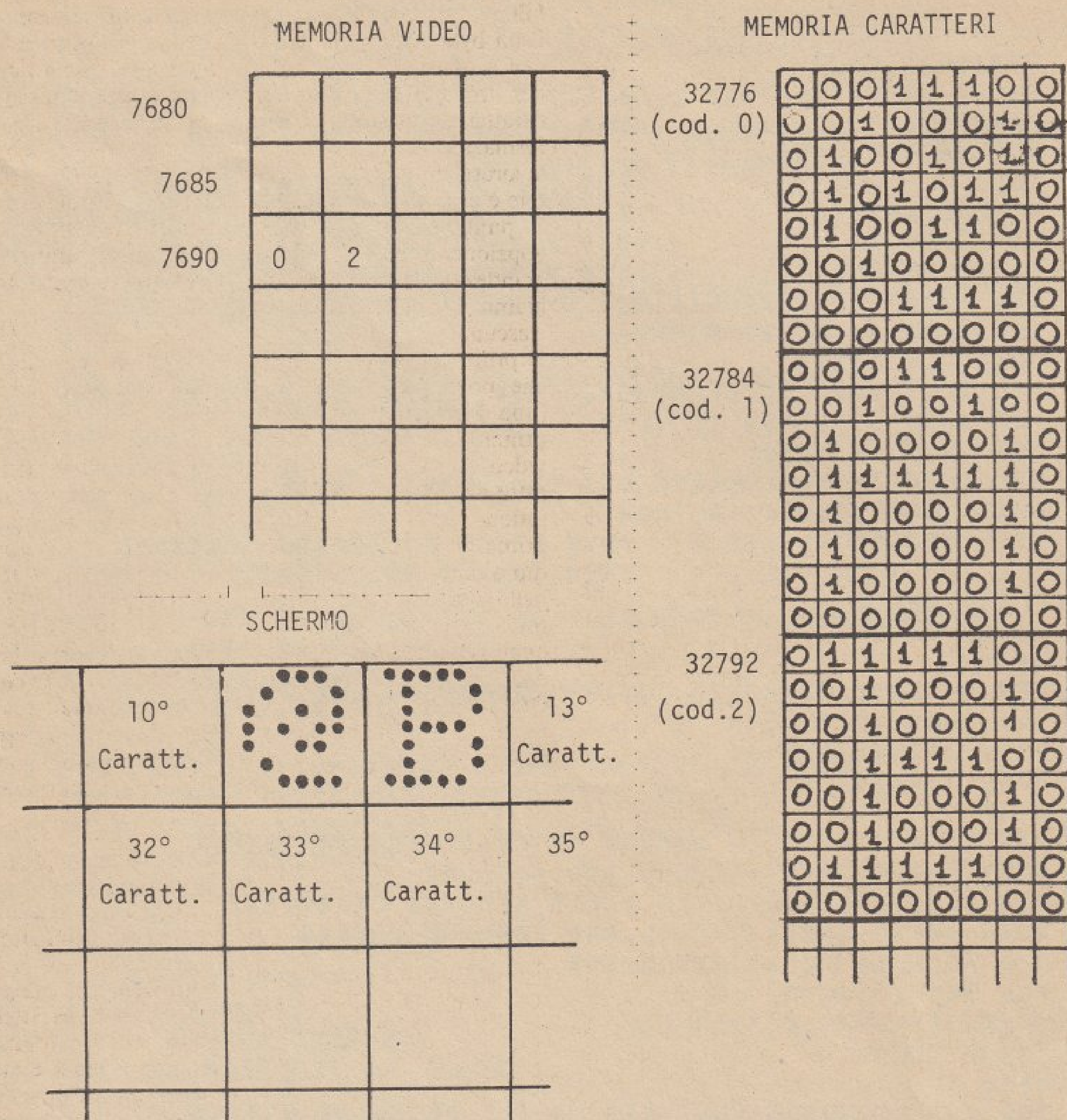
/ RUN STOP / RESTORE /
e poi:
POKE 36879,62
POKE 7690,0
Il primo POKE serve a colorare di blu lo schermo, il secondo mette a metà della prima riga il carattere con codice 0. Dovreste vedere una «@». Nella **figura 1** potete vedere le relazioni tra la memoria video, la memoria caratteri e lo schermo.
Se fate il POKE dei valori da 0 a 255 nelle prime 256 posizioni della memoria video, tutti i caratteri possibili vi saranno mostrati in ordine.
Quello che voi vedete sullo schermo non sono i 256 caratteri possibili del VIC, ma tutti i bits che sono messi a uno nei 2048 bytes che cominciano all'indirizzo definito come memoria caratteri! (256 indici che identificano i caratteri x 8 righe x 8 punti = 16384 punti = 16384 bits = 2048 bytes x 8 bits).
Dicendo al VIC che la memoria caratteri è localizzata in RAM, si può controllare a programma quali punti «accendere» o «spegnere» sullo schermo. Questa è la maniera in cui si creano i caratteri speciali.

Caratteri e doppia altezza

C'è ancora una importante osservazione da fare. Come tutti sanno, il video del VIC è fatto da 22 caratteri per 23 righe, per un totale di 506 caratteri visibili contemporaneamente. Come si fa a riempire uno schermo di 506 caratteri con solo 256 combinazioni a disposizione (visto che noi ovviamente vogliamo usare la grafica a tutto schermo)? Il problema è che i caratteri sono usati come «indici», per cui se si vuole utilizzare lo schermo in piena grafica avremo bi-

Figura 1

```
10 POKE 36879,62
20 FOR I = 5120 TO 6143 : POKE I,0 : NEXT
30 POKE 7680,0
40 POKE 36869, (PEEK (36869) AND 240) + 13
50 POKE 5120,1
60 FOR X = 1 TO 10000 : NEXT
65 POKE 36869,240
70 END
```



sogno di 506 indici.
Il trucco è di usare i caratteri a doppia altezza. Questi caratteri non cambiano la grandezza dei punti sullo schermo, cosicché ogni carattere «copre» realmente il doppio dell'area di un carattere normale. Per cui 256 indici di caratteri a doppia altezza copriranno tanto quanto 512 caratteri normali.

Utilizzo della memoria per la grafica

Per vedere le cose sotto un'altra angolazione, lo schermo del VIC è largo 176 punti (22 caratteri x 8 punti di larghezza ciascuno = 176) e alto 184 (23 righe alte 8 punti ciascuna = 184), per un totale di 32384 punti (4048 bytes). I caratteri di grandezza doppia ci danno 32768 punti (256 indici x 16 righe x 8 punti), cosicché abbiamo risolto tutti i nostri problemi. **Giusto? Sbagliato.** Il problema è la memoria che il chip del VIC può indirizzare. Come viene spiegato nel modulo di espansione, il chip «VIC» (e cioè l'integrato che «gestisce» il video, all'opposto del Computer VIC), può indirizzare solo memoria da 4096 a 8191 (da c1000 a c1FFF in esadecimale). E in questa memoria ci devono stare sia la memoria video sia la memoria caratteri. Mentre questi 4096 bytes sono sufficienti per contenere un set completo di caratteri a doppia altezza di memoria caratteri, noi abbiamo ancora bisogno di prendere 512 bytes di memoria video da questa stessa area (Ricordate? 22 righe x 23 colonne = 506 caratteri, che arrotondiamo a 512). Diciamo subito, quindi, che non sarà mai possibile avere una grafica ad alta risoluzione di 176x184 punti, ma al massimo otterremo 176x160 punti, dando 3584 bytes alla grafica e 512 alla memoria video.

Gli indirizzi di memoria utilizzabili

Abbiamo discusso finora la maggior parte delle informazioni che voi usate per fare della grafica sul VIC. Ci sono ancora alcuni dettagli tecnici e compromessi concernenti l'ammontare di grafica e di memoria che servono al BASIC. La memoria caratteri può cominciare ad una delle seguenti 4 locazioni di memoria RAM:

Programma 1: VIC con espansione da 3K
Grafica ad alta risoluzione

```
 10 PRINT"DIMOSTRAZIONE GRAFICA DEL VIC"
 10 PRINT"PC CLUB"
 30 POKE1,PEEK(55):POKE 2,PEEK(56)  :REM SALVA LA FINE CORRENTE
 40 POKE55,0                   :REM METTE LA NUOVA FINE A 4096
 45 POKE56,4096/256
 50 CLR                        :REM CLR PER AVVERTIRE IL BASIC
100 MC=4096:NR=10              :REM MEMORIA CARATTERI -NUMERO RIGHE
110 NC=22:PV=16                :REM NUMERO CARATTERI - PUNTI VERICALI
120 FS=NR*PV/2-1               :REM SCELTA FATTORE DI SCALA
200 REM AZZERA LA MEMORIA CHE DIVERRA' MEMORIA CARATTERI
210 FOR X=CM to MC+NR*NC*PV-1
220 POKE X,0 :NEXT
230 REM TUTTI UNO NELLA MEMORIA NON USATA PER FARE IL BORDO
240 FOR X=MC+NR*NC*PV TO 7679
250 POKE X,255: NEXT
260 REM SCEGLIE IL FORMATO DEI CARATTERI A 8x16
270 POKE 36867,PEEK(36867)OR 1
280 BY=PEEK(36879) AND 7     :REM LEGGI LO SFONDO CORRENTE
310 FOR X=0 TO NR*NC-1       :REM INDICI NELLA MEMORIA VIDEO
320 POKE 7680+X,X
330 POKE 38400+X,BY
340 NEXT
350 REM RIEMPIE IL RESTO CON UN CARATTERE NON UTILIZZATO
360 FOR X=NR*NC TO 511
370 POKE 7680+X,NR*NC
380 POKE 38400+X,BY
385 NEXT
390 REM DICE AL VIC II NUMERO DI CARATTERI PER RIGA
400 POKE 36866,(PEEK(36866) AND 128)+NC
410 REM CAMBIA L'INDIRIZZO DELLA MEMORIA CARATTERI
420 POKE 36869,(PEEK(36869) AND 240)+MC/1024+8
450 REM FORMULA DELLA SINUSOIDE
460 FOR X=0 TO NC*8-1
480 Y=INT(SF+SF*SIN(X/10)+1)
490 GOSUB 1000
495 NEXT
510 FOR I=1 TO 10000:NEXT:REM UN PO' PRIMA DI FINIRE
600 POKE55,PEEK(1)          :REM RIMETTE A POSTO IL LIMITE MEMORIA
610 POKE56,PEEK(2)
620 SYS(59829)        :REM RIMETTE A POSTO IL RESTO
630 END
1000 YR=Y/PV
1010 CA=INT(X/8)+INT(YR)*NC
1020 RI=(YR-INT(YR))*PV
1030 BY=MC+CA*PV+RI
1040 BI=7-(X-(INT(X/8)*8))
1050 POKE BY,PEEK(BY) OR (2↑BI)
1060 RETURN
```

4096, 5120, 6144 o 7168 (con un 12, 13, 14 o 15 negli ultimi 4 bits della locazione 36869). La memoria video può essere in una qualunque delle seguenti locazioni di memoria RAM: 4096, 4608, 5120, 5632, 6144, 6656, 7168 o 7680 (con i bits da 4 a 7 della locazione 36869 contenenti 12, 13, 14 o 15, che controllano l'allineamento di 1024 in 1024 e il bit 7 della locazione 36866 che controlla se si è allineati ai 1024 oppure ai 512).
Come vedete, la stessa locazione di memoria seleziona sia l'indirizzo della memoria video, sia l'indirizzo della memoria caratteri. Per essere più pratici diciamo che per quanto riguarda la memoria video, la selezione a 4096 viene fatta facendo il POKE di 192 (che sarebbe il valore di un 12 messo nei bits da 4 a 7) nella locazione

Programma 2 - Grafica ad alta risoluzione

```
10 PRINT"GRAFICA CON IL JOYSTICK : SENZA ESPANSIONI"
20 POKE 55,0:POKE 56,20  :REM LIMITA MEMORIA AL BASIC
40 DD=37154:P1=37151:P2=37152
50 X=10:Y=10
60 MC=5120:NR=7           :REM MEMORIA CARATTERI - NUM. RIGHE
65 NC=17:PV=16            :REM NUMERO CARATTERI - PUNTI VERICALI
80 REM AZZERA LA MEMORIA CARATTERI
85 FOR I=MC to MC+NR*NC*PV-1
90 POKE I,0 :NEXT
100 REM TUTTI UNO NELLA MEMORIA NON USATA PER FARE IL BORDO
110 FOR I=MC+NR*NC*PV TO 7679
120 POKE I,255: NEXT
130 REM SCEGLIE IL FORMATO DEI CARATTERI A 8x16
140 POKE 36867,PEEK(36867)OR 1
150 REM POKE DELLA MEMORIA VIDEO
155 BY=PEEK(36879) AND 7
160 FOR I=0 TO NR*NC-1
170 POKE 7680+I,I
172 POKE 38400+I,BY
175 NEXT
180 REM RIEMPIE IL RESTO CON UN CARATTERE NON UTILIZZATO
190 FOR I=NR*NC TO 511
200 POKE 7680+I,NR*NC
202 POKE 38400+I,BY
205 NEXT
210 REM CAMBIA L'INDIRIZZO DELLA MEMORIA CARATTERI
215 POKE 36866,(PEEK(36866) AND 128)+NC
220 POKE 36869,(PEEK(36869) AND 240)+MC/1024+8
230 REM LOOP PRINCIPALE
240 GOSUB 1010
250 YR=Y/PV
260 CA=INT(X/8)+INT(YR)*NC
270 RI=(YR-INT(YR))*PV
275 BY=MC+CA*PV+RI
280 BI=7-(X-(INT(X/8)*8))
300 POKE BY,PEEK(BY) OR (2↑BI)
310 GOTO 240
1000 REM LETTURA JOYSTICK
1010 POKE DD,127:P=PEEK(P2) AND 128
1020 J0=-(P=0)
1030 POKE DD,255:P=PEEK(P1)
1040 J1=-((P AND 8)=0)
1050 J2=-((P AND 16)=0)
1060 J3=-((P AND 4)=0)
1070 IF J0=1 THEN X=X+1
1080 IF J2=1 THEN X=X-1
1090 IF J1=1 THEN Y=Y+1
1100 IF J3=1 THEN Y=Y-1
1110 IF Y>112 THEN Y=112
1115 IF X>136 THEN X=136
1120 RETURN
```

36869, per selezionarla a 5120 il POKE è di 208 (un 13 sempre nei bits 4-7), per 6144 il POKE è di 224 (un 14) e per il 7168 il POKE è di 240 (un 15). Questo perché normalmente il bit 7 della locazione 36866 indica l'allineamento a 512. Se si volessero i valori 4608, 5632, 6656 e 7680, bisognerebbe fare anche un POKE a 36866 (POKE 36866, (PEEK (36866) OR128)).

Ai valori dei POKE sopra descritti bisogna aggiungere la selezione della memoria caratteri, semplicemente sommando un 12 (per memoria caratteri a 4096) o un

REMarks - Programma 1

30 -	Salva in 1 e 2 il limite di memoria per il programma BASIC
40 - 45	Stabilisce un nuovo limite di memoria, che è sotto la memoria caratteri
100 - 120	Mettono a posto le costanti usate in seguito MC è la locazione della memoria caratteri NR è il numero di righe NC è il numero di caratteri per riga PV è il numero di punti verticali x caratt. FS è un fattore di scala per centrare la sinusoide Notare che NR*PV è il numero di punti verticali e che NC*8 è il numero di punti orizzontali.
200 - 220	Loop che inizializza la memoria caratteri che useremo per la grafica, mettendo tutto a 0
230 - 250	Inizializza il resto della memoria caratteri mettendo tutti 1
260 - 270	Mette il formato dei caratteri a 8 x 16
280	Legge lo sfondo attuale
310 - 340	Fa il POKE dei numeri da 0 a 219 (153 per i VIC senza espansione) nella memoria video
350 - 385	Fa il POKE di un carattere non usato nel resto della memoria video
390 - 400	Cambia il numero di caratteri per riga (necessario solo per le versioni senza espansioni)
410 - 420	Muove la memoria caratteri nella RAM precedentemente preparata. (lo schermo apparirà restringersi a questo punto)
450 - 495	Disegna la sinusoide, chiamando la subroutine di plot a 1000. Se si vuole utilizzare questo programma per altra grafica, basta mettere la vostra formula per le coordinate X e Y qui mantenendo ovviamente il GOSUB1000
510 -	Aspetta un po' prima di finire
600 - 630	Rimette tutto a posto e chiude
1000 -	Viene fatta una volta solo l'operazione Y/PV (ripetuta in 2 formule)
1010 - 1060	Subroutine per trovare il bit e metterlo a uno con il POKE necessario.

13 (per 5120) o un 14 (per 6144) o un 15 (per 7168) al valore sopra indicato. Così, se per esempio volessimo la memoria video a 6656 e la memoria caratteri a 5120, dovremmo fare il POKE, sempre a 36869, di 224+13, e cioè: POKE 36869,237.
Una maniera più elegante è quella di fare il POKE del PEEK del valore che c'è, messo in AND con il valore che seleziona la memoria video e sommato con il valore che sceglie la memoria caratteri. In questo modo c'è un piccolissimo rischio (sapete dirmi quale?) ma la leggibilità del programma guadagna enormemente.
Come abbiamo visto, le memorie caratteri e video sono scelte indipendentemente una dall'altra e potrebbero occupare anche la stessa porzione di memoria. Infatti per una grafica con la massima risoluzione possibile, esse devono sovrapporsi per un poco.
Se la memoria caratteri è messa a 4096 (più realisticamente), e la memoria video è messa a 7680 (ancora più realisticamente), (facendo «POKE 36869, (PEEK (36869) AND 240) + 12'), ci sono 3584 bytes disponibili per la grafica (7680 - 4096 = 3584). Ciò ci permette di avere 22 colonne per 10 righe, oppure 176 punti orizzontali per 160 punti verticali (ogni riga è alta 16 punti e larga 8 punti x 22 caratteri il tutto usando i caratteri doppi). Questo però non lascia spazio per programmi BASIC in un VIC senza espansioni di memoria. Se volete spazio per un programma BASIC che occupi non più di 1K, spostate la memoria caratteri a 5120 e mantenete la memoria video a 7680, con il comando:

POKE 36869, (PEEK (36869) AND 240) + 13

Questa configurazione vi dà 2560 bytes per dati di grafica e una griglia di 176 punti (22 caratteri) per 112 (7 caratteri a doppia altezza) usando 2464 bytes. Questa non è una area veramente quadrata su cui disegnare, per cui sarebbe bene cambiare anche il numero di caratteri per riga da 22 a 17:

POKE 36866, (PEEK (36866) AND 128) + 17

Questo ci dà 136 punti orizzontali per 144 punti verticali. Notare, ancora, che tutta questa discussione si applica ad un VIC senza espansione o ad un VIC con una espansione di 3K di memoria.
Esiste una ulteriore complicazione per i VIC con più di 8K. Per questi sistemi, la memoria video parte (normalmente) all'indirizzo 4096, e il programma BASIC comincia a 4608.
Per usare il tipo di grafica di cui stiamo parlando, l'inizio del programma BASIC deve essere spostato sopra l'area usata per le memorie caratteri e video (sopra 8191). Sotto le giuste circostanze, esso può essere fatto dal programma BASIC che sta girando, ma è molto più semplice fare questo spostamento prima di caricare il programma. Vedremo come fare quando descriveremo le modifiche da fare ai programmi se avete questa espansione.

Qualche esempio

Provate a far girare questo breve programma **figura 2** che vi mostrerà alcuni principi fondamentali di questo tipo di grafica.
Guardate cosa è apparso in alto a sinistra sullo schermo. Un punto è stato acceso nella prima riga. La linea 10 del programma colora di blu lo schermo. La linea 20 pulisce tutta la memoria RAM da 5120 a 6143 (diverrà la nuova memoria caratteri). La linea 30 mette un codice carattere di 0 nella memoria video a 7680 (normalmente sarebbe una @). La linea 40 cambia il puntatore alla memoria caratteri da ROM a RAM, indirizzo 5120. La linea 50 crea un nuovo carattere utilizzando solo il primo degli otto bytes che definiscono il carattere con codice 0, per tutte le altre linee del carattere sono «spente». Questa linea 50 mette a 1 il bit nella posizione 0 (quello più a destra nel byte) per cui vedremo il punto più a destra della prima riga del primo carattere accendersi. La linea 60 fa in modo che il VIC aspetti un po' prima di mostrare «READY» e finire la dimostrazione.
Provate ora a codificare diversi valori nella riga 50, tenendo presente che i valori variano da 0 (nessun punto

Figura 3

Numero dei bit / Valori binari		
7 6 5 4 3 2 1 0		
0 0 0 0 0 0 0 1	=	1
0 0 0 0 0 0 1 0	=	2
0 0 0 0 0 1 0 0	=	4
0 0 0 0 1 0 0 0	=	8
1 0 0 0 0 0 0 0	=	128
1 0 0 0 0 0 0 1	=	129
1 1 1 1 1 1 1 1	=	255

Locazioni di memoria utili per la grafica ad alta risoluzione

43,44	Puntatore all'inizio del programma BASIC. Dice al BASIC il numero di pagina di dove comincia il programma. Numero di pagina = Indirizzo/256 Per ricavare l'indirizzo bisogna fare: (contenuto locazione 44)x256+(conten. locaz. 43). Per l'espansione di 8K: normalmente 1,18; cambiare a 1,32 per far cominciare il programma a 8192
45,46	Puntatore fine programma BASIC, inizio Variabili
47,48	Puntatore fine variabili, inizio matrici
49,50	Puntatore fine matrici
51,52	Puntatore fine stringhe (si muovono verso il basso)
55,56	Limite della memoria al BASIC
57,58	Numero della Linea di programma corrente
59,60	Numero della Linea di programma precedente
643,644	Limite della memoria al BASIC
641,642	Puntatore all'inizio del programma BASIC. Contiene il numero di pagina. Con l'espansione a 8K contiene 0,18; cambiare a 0,32 per 8192
648	Puntatore memoria video, deve essere modificato solo con l'espansione di 8K (metterlo a 30)
	Inizio reale del programma BASIC
1024	Con espans. 3K
4096	Senza espansioni
4608	Con sespans. 8K
	Inizio MEMORIA VIDEO standard:
4096	Con espansione da 8K
7680	Senza espansioni o con espansione da 3K
4096	o Indirizzi possibili per la memoria caratteri
5120	o
6144	o
7168	
8192	Primo indirizzo della espansione da 8K: se si sposta il programma a questo indirizzo, bisogna che il primo byte sia a 0
36869	Puntatore alla memoria caratteri e video. Per il VIDEO: POKE di 192 per puntare 4096; 224 per puntare 6144; 208 per puntare 5120; 240 per puntare 7168. Per CARATTERI: sommare al valore del POKE per il video: 12 per puntare 4096; 14 per puntare 6144; 13 per puntare 5120; 15 per puntare 7168.
36866	Per cambiare il numero di caratteri per riga NC = numero caratteri: (PEEK (36866)AND128)+NC Per cambiare allineamento della memoria video ai 512 o ai 1024(Bit 7 a 1 per 512). Normalmente: VIC senza espansione = allineamento a 512; VIC con 3K espansione = allineamento a 512; VIC con 8K espansione = allineamento a 1024.
36867	Per mettere il format dei caratteri a 8x16 (PEEK(36867) OR 1)
36879	Colora lo schermo
32768	Locazione della memoria caratteri standard
38400	Locazioni dei colori (in parallelo con la memoria video)
37888	Locazioni dei colori (espansione 8K)
59829	Routine che rimette a posto gli indirizzi per tornare allo standard: va chiamata col SYS(59829)

acceso) a 255 (tutti i punti accesi).
Per allargare la vostra comprensione, fate le seguenti modifiche al programma precedente e fatelo girare:

30 U=O : FOR J=7680 TO 7701 : POKE J U : U = U+1 : NEXT
50 POKE 5128,1

Vediamo ora come si lavora in grafica con i caratteri a grandezza doppia. Provate a battere:

POKE 36879,62
POKE 36867, (PEEK (36867) OR 1)

Non stupitevi e andate avanti «al buio»:

POKE 7690,0

Per rimettere tutto a posto battere:

POKE 36867, (PEEK (36867) AND 254)

Insieme a della «spazzatura», dovreste vedere due caratteri nel mezzo della prima riga: una «@» sopra una «A». Il secondo POKE ha cambiato la matrice dei caratteri del VIC da 8 linee di 8 punti a 16 linee di 8 punti per ogni carattere. Per cui il VIC questa volta va a leggersi i primi 16 bytes della memoria caratteri per costruire il carattere con codice 0. Dato che i secondi 8 bytes identificano una «A» (i primi otto li abbiamo già visti, sono una «@»), noi vediamo entrambi i caratteri. In questa maniera se facciamo il POKE dei valori da 0 a 253 nella memoria video, avremo lo schermo occupato in tutte le sue 506 posizioni.

Il calcolo delle coordinate X e Y

Ora che sappiamo come accendere o spegnere un punto sullo schermo cambiando un bit nella memoria caratteri, dobbiamo fare in modo che il nostro programma prenda delle coordinate X (orizzontale, da 0 a 160) e Y (verticale, da 0 a 176) e le trasformi nel corrispondente indirizzo del byte, dicendoci anche quale bit in quel byte identifica il nostro punto. Le seguenti formule sono tratte dal manuale del VIC:

Carattere
CA = INT(X/8)+INT(Y/PV)*NC
Riga del carattere
RI = (Y/PV–INT(Y/PV)*PV
Byte
BY = MC+CA*PV+RI
Bit
BI = 7–(X–(INT(X/8)*8))

Per queste formule, X rappresenta la coordinata orizzontale da sinistra a destra e Y la coordinata verticale dall'alto verso il basso. NC è il numero di caratteri per riga, che vedremo più avanti, PV è il numero di righe di punti per carattere (8 per i caratteri a grandezza normale), MC è l'indirizzo della memoria caratteri.
Per mettere a 1 qualunque bit con le coordinate X e Y, dopo le precedenti formule fare:

POKE BY,PEEK(BY)CR(2↑BI)

L'elevazione a potenza serve a fare in modo di ottenere dei valori binari «interi», e cioè che identifichino un singolo bit. Questo è possibile solo se i valori sono in potenza di due. Il risultato sarà: 0,1,2,4,8,16...ecc.
Notare anche che queste formule partono dal presupposto che si sia precedentemente riempita la memoria video con valori crescenti da 0 al numero di caratteri che avete a disposizione (esempio: 153 per un VIC senza espansioni, 220 per un VIC con espansioni).

Il programma 1

Il primo programma usa la grafica ad alta risoluzione per disegnare una sinusoide. Se voi gli date un'occhiata, troverete che buona parte del programma serve a mettere a posto le cose per la grafica, e che le linee da 460 a 490 sono quelle che creano il disegno da mostrare sul video (una semplice curva sinusoidale). Il programma come listato è per un VIC con una espansione di memoria di 3K. Se avete un VIC senza espansioni, cambiate le seguenti linee e rimuovete tutte le REMs. Ciò vi darà un campo di 136 per 144 punti:

45 POKE 56,5120/256
100 MC=5120:NR=7
110 NC=17:PV=16

Se invece avete una espansione da 8K dovete rimuovere le righe 40, 45 e 50, dato che la fine della memoria è sopra la memoria video. Dovrete anche battere i seguenti comandi in modo «diretto» (non a programma) prima di caricare il programma BASIC. Queste istruzioni spostano la memoria video dove essa è nella versione standard (senza espansioni) e cambiano l'inizio del programma BASIC in modo tale che sia sopra la memoria video. Questo vi permette di usare la memoria da 4096 a 7680 per le memorie video e caratteri.

POKE 36866,150:POKE 36869,240 :
POKE 648,30 CLR
POKE 43,1 : POKE 44,32
POKE 642,32
POKE 8192,0
CLR : NEW

Il primo POKE mette la selezione dell'allineamento della memoria video a 512 (per poterla poi mettere a 7680), il secondo mette la memoria caratteri a 4096 e quella video a 7168+512, il terzo dice il numero di pagina della memoria video (30x256 = 7680). Voi dovete premere il tasto CLR/HOME per pulire lo schermo dopo aver battuto la prima riga. Ciò dirà al BASIC che avete cambiato la locazione della memoria video. Il POKE a 43,44 e 642 informano il BASIC che deve mettere il programma sopra la memoria video, e cioè a 8192 (32x256 = 8192), mentre il POKE a 8192 mette a 0 la prima locazione di memoria utile per il programma BASIC: è un passo obbligatorio quando si sposta l'inizio del programma.

Realizzare nuovi caratteri

Questa è una procedura che potrete anche inserire come routine nei vostri programmi che vi permetterà di utilizzare le possibilità di alta risoluzione del vostro Vic 20. Basta che abbiate la versione di base dei Vic, cioè i 3.5 Kb di memoria utilizzabile (o «memoria viva»). Tale procedura fa uso della possibilità di usare la «memoria morta» partendo dall'indirizzo N. 36869.
Provate prima di scrivere l'intera procedura, a digitare: «POKE 36869, 220 RETURN», i caratteri da questo punto in poi, saranno svincolati e dipenderanno solo dal contenuto della «memoria viva».
Per tornare ai caratteri da voi abitualmente usati sul vostro Vic, basterà premere Run o Stop o Restore.
Quindi, avendo a disposizione un bel po' di «memoria viva», potremo contare su un numero elevato di caratteri in più, e tutti da noi scelti, creati e voluti.
Come fare per creare dei caratteri nuovi?
Nel caso non fosse a voi noto, un video è un insieme di punti e per far scrivere una parola devo «accendere» alcuni punti, ed altri lasciarli «spenti» come tante piccole lampadine.
O, meglio dire, devo crearmi una «matrice» di punti, sapendo che il mio video è una matrice quadrata di 256 punti per le righe, per altrettanti 256 per le colonne, cioè da 0 a 255.
Ponendo 0 come valore ad un punto (X, Y) avrò che tale punto sarà «spento», mentre con il valore ad 1, avrò che tale punto sarà «acceso». Quindi il carattere viene creato, (forse sarebbe meglio dire solo visualizzato) in base ai punti «accesi».

Ogni carattere è formato da una matrice di 8 righe x 8 colonne.

Listato - Nuovi Caratteri

```
10 REM.VIC 20
20 DIMA(7):N=0:GOTO10000
25 FORX=0TO7:A(X)=0:NEXT
26 I=0:J=0:GOTO 1500
30 POKE 7600+22*I+J,160
35 POKE 38400+22*I+J,6
40 GETA$
50 IFA$=CHR$(13)THENA(I)=A(I)+2↑(7-J)
55 POKE 7680+22*I+J,160
56 J=J+1
60 IFA$=" "THENPOKE7680+22*I+J,32
65 J=J+1
70 IF J>7 THEN I=I+1:J=0
80 IFI>7THEN110
90 POKE7680+22*I+J,32
100 GOTO30
110 FORX=0TO7
120 POKE6144+N*8+X,A(X)
130 NEXT
140 INPUT "UN ALTRO CARATTERE";R$
150 IFR$="SI"THENN=N+1:GOTO20
160 IF R$="NO"THEN180
170 GOTO140
180 PRINT:POKE36869,254:N=N+1
190 END
1500 PRINT "                L0
1510 PRINT "                 L1
1520 PRINT "                 L2
1530 PRINT "                 L3
1540 PRINT "                 L4
1550 PRINT "                 L5
1560 PRINT "                 L6
1570 PRINT "                 L7
1580 PRINT "[][][][][][][]"
1590 PRINT "01234567"
1600 GOTO30
10000 FORX=6144TO7680
10010 POKEX,0:NEXT
10020 GOTO20
```

Simulare l'istruzione PLOT sul C.64

Questo programma, di **Paolo Peranzoni** di Padova, colma una grossa lacuna del BASIC del **C-64** e cioè l'istruzione PLOT/ DRAW. È un programma interamente in linguaggio macchina (quindi OK come occupazione di memoria e velocità), è originale ed è ad alto contenuto tecnico. La documentazione è abbastanza completa, l'uso è relativamente facile (una volta capite le regole di «attivazione» basta dare un SYS... e le coordinate) ed è veramente molto veloce.
Il programma in linguaggio macchina per COMMODORE 64 ha lo scopo di simulare l'istruzione PLOT, purtroppo assente dal BASIC Commodore, viene usato il comando SYS per chiamare la routine in I.m., per cui, dopo aver posto PL=49292 (vedi programma BASIC dimostrativo, linea 80), la sintassi del comando è:

SYSPL, X, Y, per disegnare il punto coordinate (X, Y), dove X ed Y stanno per qualunque espressione algebrica calcolabile in BASIC oppure

SYSPL, X, Y, X1, Y1,... per disegnare più punti oppure

SYSPL, X, Y TO X1, Y1 per disegnare il segmento che unisce (X, Y) con (X1, Y1) oppure

SYSPL, X, Y TO X1, Y1 TO X2, Y2 TO... per disegnare più segmenti consecutivi.

Il comando **SYSPL TO X,** Y... traccia un segmento dall'ultimo punto precedentemente disegnato fino ad (X Y); se non è ancora stato disegnato nessun punto, oppure se è stato azzerato il flag di abilitazione tracciamento segmenti (locaz. $FE), il comando equivale a SYSPL, X, Y...

Si possono liberamente mescolare le due modalità (singoli punti e segmenti) in una stessa istruzione, purché ovviamente non si ecceda la lunghezza di 80 caratteri di una linea BASIC.

Prima di poter usare l'istruzione SYSPL è necessario:

1) Caricare la routine in I.m. in memoria: ciò è realizzato dal programma BASIC mediante la lettura di una serie di DATA decimali.

2) Pulire la pagina grafica prescelta e riempire la mappa colore; ciò è effettuato da una routine in I.m., che viene chiamata dal BASIC mediante il comando SYSCL (vedi listato, linee 70, 110 e 140).

3) Predisporre il chip VIC II per la grafica: ciò è ottenuto con le linee 80 e 90 del programma BASIC: non ho implementato questa routine in I.m. dato che non ci sono problemi di velocità.

Il programma in I.m. fa largo uso di routines del BASIC; per individuare i punti di ingresso di tali routines mi sono servito del libro VIC REVEALED di Nick Hampshire, dato che il BASIC dei due computer è uguale, a parte le zone di memoria; purtroppo il libro è ricco di inesattezze che rendono faticoso un corretto uso delle routine elencate. **(W.P.)**

Note e commenti ai programmi - Istruzione PLOT

Il programma in I.m. si sviluppa secondo il diagramma di flusso in figura 1 nel quale X, Y indicano le coordinate del punto da disegnare (o dell'inizio del segmento nel caso di SYSPL TO) e X1, Y1 le coordinate della fine del segmento; N rappresenta il numero di passi necessari per il tracciamento, mentre ΔX' e ΔY' rappresentano gli incrementi delle due variabili ad ogni passo.
Le diverse quantità da memorizzare vengono sistemate in blocchi da 5 bytes ciascuno, denominati ACC3, ACC4, ecc., in analogia con ACC1 e ACC2 (primo e secondo float. acc.). Gli indirizzi di partenza e i contenuti sono:

ACC3 da $ØØ 57	contiene N prima della trasf. in intero
ACC4 da $ØØ 5C	contiene ΔY e poi ΔY'
ACC5 da $Ø 2A7	contiene X dell'ultimo punto
ACC6 da $Ø 2AC	contiene Y
ACC7 da $Ø 2B1	contiene ΔX e poi ΔX'

Per il contatore dei passi si utilizzano poi le locazioni $4B e $4C; altre locazioni che non interferiscono col BASIC sono usate per passare parametri.
La locazione $Ø 2 è usata per il codice colore (semi-byte alto = colore punto, semi-byte basso = colore sfondo).
I vari arrotondamenti all'intero si rendono necessari sia per garantire una corretta rappresentazione cartesiana (per inciso: l'origine è in basso a sinistra, e gli assi orientati come di consueto), sia per evitare irregolarità nel tracciamento dei segmenti.
Il diagramma di flusso della subroutine PLOT è mostrato in figura 2.
L'algoritmo usato riprende noti metodi proposti, in BASIC, anche dal manuale COMMODORE, con qualche miglioramento legato all'uso delle funzioni logiche. Il puntatore alla memoria di schermo è costituito dalle locazioni $22 e $23.
Il programma BASIC oltre a caricare la routine in I.m., costituisce una dimostrazione dell'uso di questa istruzione; disegna una serie di quadrati concentrici, e poi una figura di Lissajus. La scelta del colore (codice 2Ø 7 = punti grigio-medio su sfondo grigio-chiaro) è legata all'uso di un monitor monocromatico a fosfori verdi, dove offre risultati soddisfacenti senza dover modificare luminosità e contrasto rispetto alla pagina testo; su un televisore a colori si possono usare altre combinazioni, tenendo presente però che spesso insorgono aberrazioni cromatiche.
Ultima annotazione: la somma del DATA (utile per un controllo di esattezza) deve dare 50830.

Push Over per VIC 20

Il listato di «Push Over», gioco di strategia di scacchiera viene proposto dal fiorentino **Simone Marinai.**
«Push Over» è un gioco da scacchiera (5x5), che richiede di allineare quattro pedine del proprio colore in orizzontale o verticale (non diagonale), per vincere. La differenza con i normali «filetti» sta tutta nel fatto che è vietato deporre pedine direttamente sulla scacchiera, ma bisogna al contrario appoggiarle all'esterno di essa, intorno, facendo poi scorrere la fila di pedine fino a far entrare la propria (da qui il nome del gioco). Esempi e figure sono stati discussi su Personal Computer n. 5/83, ma il listato del nostro amico fiorentino dovrebbe aiutarvi e chiarire ogni dubbio. Buon lavoro! **(L.F.)**

(Il listato è a pag. 36)

Fig. 1 - Diagramma di flusso del programma L.M. disassemblato

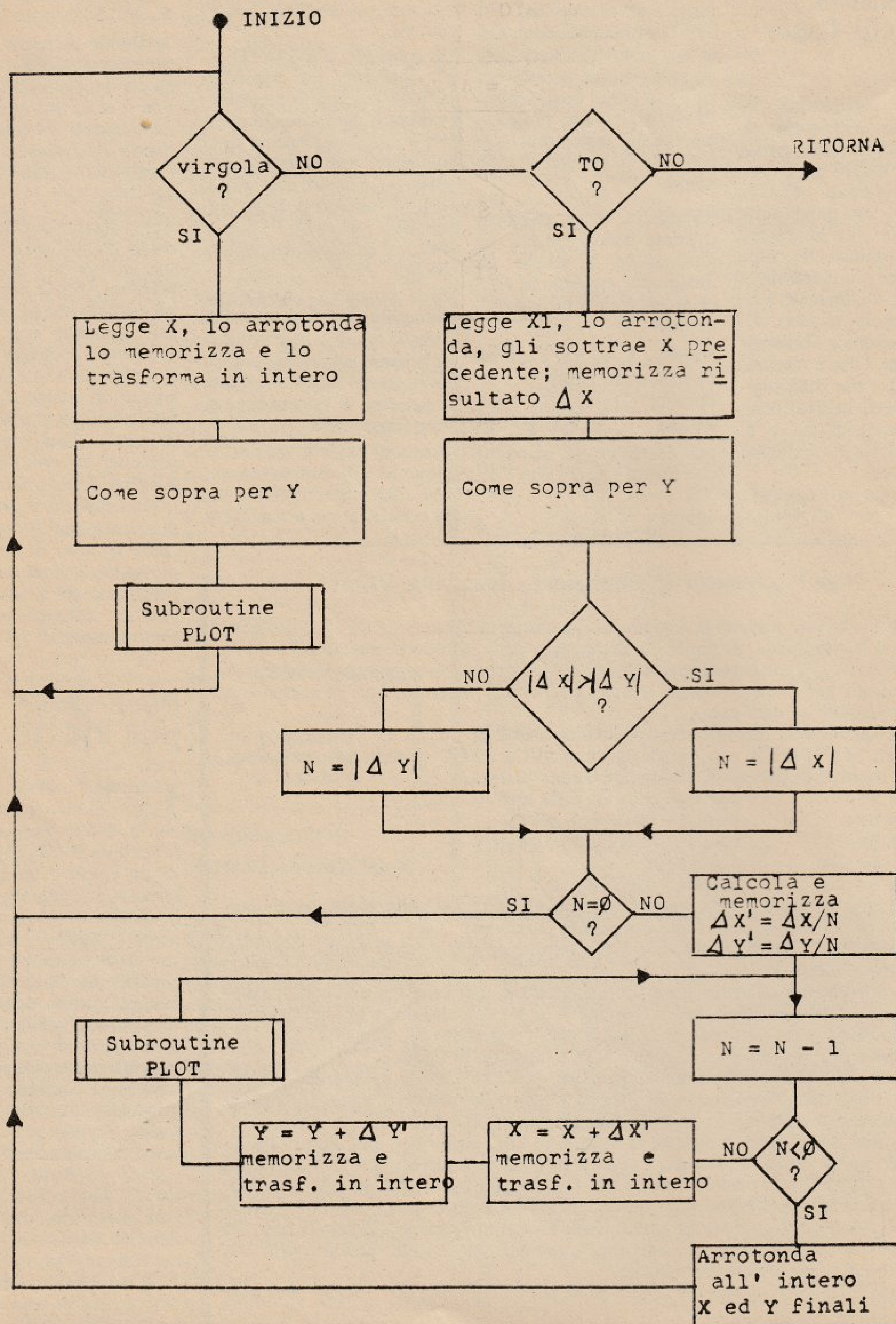
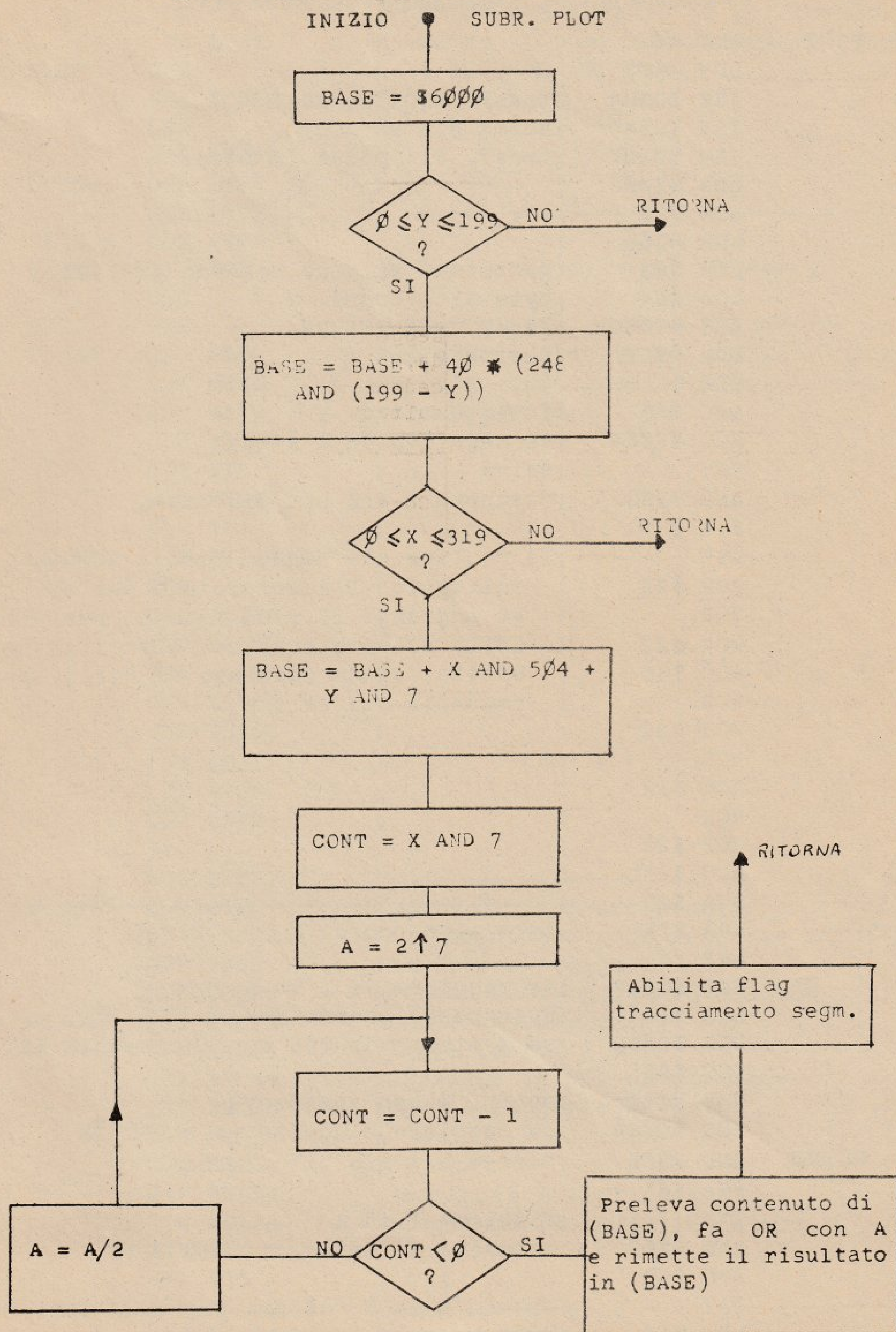


Fig. 2 - Diagramma di flusso della subroutine PLOT.



Programma in L.M. disassemblato - Istruzione PLOT

```
CØØØ  PLOT    LDX #$AC
CØØ2          LDY #$Ø2
CØØ4          JSR $BBD4    ;trasf. ACC1 in ACC6
CØØ7          JSR $B849    ;somma Ø.5 ad ACC1
CØØA          JSR $B1BF    ;trasf. fl. point in intero
CØØD          LDA #$ØØ     ;
CØØF          STA $22      ;
CØ11          LDA #$6Ø     ;
CØ13          STA $23      ;puntatore origine schermo grafico
CØ15          LDA $64      ;byte alto coord. Y
CØ17          BNE $CØ64    ;se ≠Ø va a RETURN
CØ19          LDA #$C7     ;decimale 199
CØ1B          SEC
CØ1C          SBC $65      ;inverte coord. Y
CØ1E          BCC $CØ64    ;se negativo va a RETURN
CØ2Ø          TAY          ;salva A in Y
CØ21          AND #$F8     ;maschera i tre bit più bassi
CØ23          STA $65
CØ25          LSR          ;la routine che segue, fino a $CØ39,
CØ26          ROR $22      ;  moltiplica l'accumulatore per 4Ø
CØ28          ROR          ;  ed aggiunge il risultato al puntatore
CØ29          ROR $22      ;  dello schermo grafico; è scritta per
CØ2B          ADC $65      ;  esteso, senza fare uso di cicli, per
CØ2D          ROR          ;  renderla più veloce
CØ2E          ROR $22      ;
CØ3Ø          ROR          ;
CØ31          ROR $22      ;
CØ33          ROR          ;
CØ34          ROR $22      ;
CØ36          ADC $23      ;
CØ38          STA $23      ; ---------------------------------
CØ3A          LDA $15      ;byte alto coord. X
CØ3C          LSR
CØ3D          BNE $CØ64    ;se maggiore di 1 va a RETURN
CØ3F          LDA $14      ;byte basso coord. X
CØ41          BCC $CØ49    ;se X minore di 256 non incrementa $23
CØ43          INC $23
CØ45          CMP #$4Ø     ;confr. X con 32Ø(dec.)
CØ47          BCS $CØ64    ;se maggiore o uguale va a RETURN
CØ49  NOINC   AND #$F8     ;maschera i tre bit più bassi
CØ4B          STA $65
CØ4D          TYA          ;richiama Y in A
CØ4E          AND #$Ø7     ;maschera i cinque bit più alti
CØ5Ø          ADC $65
CØ52          TAY          ;trasf. A in Y (offset per ind. indiretto)
CØ53          LDA $14      ;byte basso X
```

```
CØ55            AND #$Ø7      ;maschera i cinque bit più alti
CØ57            TAX           ;routine per settare il bit corrispon_
CØ58            LDA #$8Ø      ;  dente al punto da accendere (fino
CØ5A  LOOP      DEX           ;  a RETURN  escluso)
CØ5B            BMI $CØ6Ø     ;
CØ5D            LSR           ;
CØ5E            BNE $CØ5A     ;
CØ6Ø  LPEND     ORA ($22),Y   ;
CØ62            STA ($22),Y   ; --------------------------------
CØ64  RETURN    LDX #$FF
CØ66  RET2      STX $FE       ;abilita tracciamento segmenti
CØ68            CLC
CØ69  RET1      RTS           ;ritorna alla routine chiamante
CØ6A  CLEAR     LDA #$ØØ      ;questa routine (fino a $CØ8B ) pulisce
CØ6C            LDX #$2Ø      ;  lo schermo grafico e predispone la
CØ6E            LDY #$6Ø      ;  mappa colore
CØ7Ø            CLC           ;
CØ71  BIS       STY $15       ;
CØ73            LDY #$ØØ      ;
CØ75            STY $14       ;
CØ77  LOOP1     STA ($14),Y   ;
CØ79            INY           ;
CØ7A            BNE $CØ77     ;
CØ7C            INC $15       ;
CØ7E            DEX           ;
CØ7F            BNE $CØ77     ;
CØ81            BCS $CØ66     ;
CØ83            LDA $Ø2       ;colori sfondo e punto
CØ85            LDX #$Ø4      ;
CØ87            LDY #$5C      ;
CØ89            SEC           ;
CØ8A            BCS $CØ71     ;---------------------------------
CØ8C  START     LDY #$ØØ      ;punto di ingresso routine
CØ8E            LDA ($7A),Y   ;analizza carattere seguente SYSFL
CØ9Ø            CMP #$2C      ;carattere 'virgola'
CØ92            BNE $CØBF     ;se non è 'virgola' va a NOCOM
CØ94  SINGLE    JSR $ØØ73     ;legge un carattere
CØ97            JSR $AD8A     ;legge coord. X
CØ9A            JSR $B849     ;aggiunge .5 per arrotondare
CØ9D            JSR $BCCC     ;prende la parte intera
CØAØ            LDX #$A7      ;
CØA2            LDY #$Ø2      ;
CØA4            JSR $BBD4     ;trasferisce ACC1 in ACC5
CØA7            JSR $B1AA     ;trasforma fl.point in intero
CØAA            STY $14
CØAC            STA $15
CØAE            JSR $AEFD     ;verifica virgola
CØB1            JSR $AD8A     ;legge coord. Y
```

Programma in L.M. disassemblato - Istruzione PLOT

```
CØB4          JSR $B849     ;aggiunge .5
CØB7          JSR $BCCC     ;parte intera
CØBA          JSR $CØØØ     ;va alla subr. PLOT
CØBD          BCC $CØ8C     ;torna a  START
CØBF  NOCOM   CMP #$A4      ;codice BASIC comando  TO
CØC1          BNE $CØ69     ;se non è 'TO' va a RET1 (ritorna)
CØC3          LDA $FE       ;flag abilitazione tracciamento segm.
CØC5          BEQ $CØ94     ;se =Ø va a  SINGLE (ignora TO)
CØC7          JSR $ØØ73     ;legge un carattere
CØCA          JSR $AD8A     ;legge coord. X
CØCD          JSR $B849     ;aggiunge .5
CØDØ          JSR $BCCC     ;parte intera
CØD3          LDA #$A7
CØD5          LDY #$Ø2
CØD7          JSR $B85Ø     ;ACC5-ACC1 in ACC1
CØDA          LDX $B1
CØDC          LDY $Ø2
CØDE          JSR $BBD4     ;trasferisce ACC1 in ACC7
CØE1          JSR $AEFD     ;verifica virgola
CØE4          JSR $AD8A     ;legge coord. Y
CØE7          JSR $B849     ;+.5
CØEA          JSR $BCCC     ;parte intera
CØED          LDA #$AC
CØEF          LDY #$Ø2
CØF1          JSR $B85Ø     ;ACC6-ACC1 in ACC1
CØF4          JSR $BBC7     ;ACC1 in ACC4
CØF7          LSR $66       ;valore assoluto di ACC1
CØF9          LDA #$B1
CØFB          LDY #$Ø2
CØFD          JSR $BA8C     ;trasferisce ACC7 in ACC2
C1ØØ          LDA $6A       ;primo byte mantissa ACC2
C1Ø2          AND #$7F      ;maschera il bit più alto
C1Ø4          STA $6A
C1Ø6          LDA #$69
C1Ø8          LDY #$ØØ
C1ØA          JSR $BC5B     ;confronta ACC1 con ACC2
C1ØD          ASL
C1ØE          BCC $C117     ;se ACC1 è maggiore di ACC2 va a MAG
C11Ø          LDA #$69
C112          LDY #$ØØ
C114          JSR $BBA2     ;trasferisce ACC2 in ACC1
C117  MAG     JSR $B1AA     ;trasforma fl.point in intero
C11A          STY $4B       ;contatore passi tracciamento
C11C          STA $4C       ;
C11E          BNE $C126     ;se maggiore di Ø va a DRAW
C12Ø          TAX
C121          BNE $C126     ;
```

```
C123              JMP $CØ8C      ;torna a START
C126    DRAW      JSR $B391      ;trasforma intero in fl. point
C129              JSR $BBCA      ;trasferisce ACC1 in ACC3
C12C              LDA #$B1
C12E              LDY #$Ø2
C13Ø              JSR $BBØF      ;ACC7/ACC1 in ACC1
C133              LDX #$B1
C135              LDY #$Ø2
C137              JSR $BBD4      ;trasferisce ACC1 in ACC7
C13A              LDA #$57
C13C              LDY #$ØØ
C13E              JSR $BBA2      ;trasferisce ACC3 in ACC1
C141              LDA #$5C
C143              LDY #ØØ
C145              JSR $BBØF      ;ACC4/ACC1 in ACC1
C148              JSR $BBC7      ;ACC1 in ACC4
C14B    LOOP2     LDA $4B        ;
C14D              BNE $C153      ;
C14F              DEC $4C        ;decrementa contatore passi (byte alto)
C151              BMI $C187      ;se il trcciamento è finito va a START1
C153              DEC $4B        ;decrementa contatore passi (byte basso)
C155              LDA #$B1
C157              LDY #$Ø2
C159              JSR $BBA2      ;trasferisce ACC7 in ACC1
C15C              LDA #$A7
C15E              LDY #$Ø2
C16Ø              JSR $B85Ø      ;ACC5-ACC1 in ACC1
C163              LDX #$A7
C165              LDY #$Ø2
C167              JSR $BBD4      ;trasferisce ACC1 in ACC5
C16A              JSR $B849      ;+.5
C16D              JSR $B1AA      ;trasforma fl.point in intero
C17Ø              STY $14
C172              STA $15
C174              LDA #$5C
C176              LDY #$ØØ
C178              JSR $BBA2      ;trasferisce ACC4 in ACC1
C17B              LDA #$AC
C17D              LDY #$Ø2
C17F              JSR $B85Ø      ;ACC6-ACC1 in ACC1
C182              JSR $CØØØ      ;va alla subr. PLOT
C185              BCC $C14B      ;va a LOOP2
C187    START1    LDA #$A7       ;questa routine(fino alla fine)arrotonda
C189              LDY #$Ø2       ; all'intero il contenuto di ACC5 e di
C18B              JSR $BBA2      ; ACC6 , che contengono le coordinate
C18E              JSR $B849      ; X ed Y dell'ultimo punto disegnato
C191              JSR $BCCC      ;
C194              LDX #$A7       ;
C196              LDY #$Ø2       ;
```

Programma in L.M. disassemblato - Istruzione PLOT

```
C198        JSR $BBD4     ;
C19B        LDA #$AC      ;
C19D        LDY #$Ø2      ;
C19F        JSR $BBA2     ;
C1A2        JSR $B849     ;
C1A5        JSR $BCCC     ;
C1A8        LDX #$AC      ;
C1AA        LDY #$Ø2      ;
C1AC        JSR $BBA2     ;
C1AF        JMP $CØ8C     ;--------- FINE ROUTINE L.M. ----------
```

Listato programma Basic - Istruzione PLOT

```
Ø REM ----- PLOT -----
1 DATA 162,172,16Ø,2,32,212,187,32,73,184,32,191,177,169,Ø,133,
      34,169,96,133,35
3 DATA 165,1ØØ,2Ø8,75,169,199,56,229,1Ø1,144,68,168,41,248
5 DATA 133,1Ø1,74,1Ø2,34,1Ø6,1Ø2,34,1Ø1,1Ø1,1Ø6,1Ø2,34,1Ø6,1Ø2,
      34,1Ø6,1Ø2,34
7 DATA 1Ø1,35,133,35,165,21,74,2Ø8,37,165,2Ø,144,6,23Ø,35
9 DATA 2Ø1,64,176,27,41,248,133,1Ø1,152,41,7,1Ø1,1Ø1,168,165,2Ø
11 DATA 41,7,17Ø,169,128,2Ø2,48,3,74,2Ø8,25Ø,17,34,145,34,162,
      255,134,254,24,96
12 REM -----CLEAR-----
13 DATA 169,Ø,162,32,16Ø,96,24,132,21,16Ø,Ø,132,2Ø,145,2Ø,2ØØ,
      2Ø8,251
15 DATA 23Ø,21,2Ø2,2Ø8,246,176,227,165,2,162,4,16Ø,92,56,176,229
16 REM
17 REM ----- START -----
18 REM
19 DATA 16Ø,Ø,177,122,2Ø1,44,2Ø8,43,32,115,Ø,32,138,173,32,73,184,
      32,2Ø4,188
21 DATA 162,167,16Ø,2,32,212,187,32,17Ø,177,132,2Ø,133,21,32,253,
      174,32,138,173
23 DATA 32,73,184,32,2Ø4,188,32,Ø,192,144,2Ø5,2Ø1,164,2Ø8,166,165,
      254,24Ø,2Ø5
25 DATA 32,115,Ø,32,138,173,32,73,184,32,2Ø4,188,169,167,16Ø,2,
      32,8Ø,184,162,177
27 DATA 16Ø,2,32,212,187,32,253,174,32,138,173,32,73,184,32,2Ø4,
      188,169,172,16Ø
29 DATA 2,32,8Ø,184,32,199,187,7Ø,1Ø2,169,177,16Ø,2,32,14Ø,186,
      165,1Ø6,41,127
31 DATA 133,1Ø6,169,1Ø5,16Ø,Ø,32,91,188,1Ø,144,7,169,1Ø5,16Ø,Ø,
      32,162,187
32 REM ----- MAG -----
```

Listato programma Basic - Istruzione PLOT

```
33 DATA 32,170,177,132,75,133,76,208,6,170,208,3,76,140,192,32,
     145,179,32,202,187
35 DATA 169,177,160,2,32,15,187,162,177,160,2,32,212,187,169,87,
     160,0,32,162,187
37 DATA 169,92,160,0,32,15,187,32,199,187,165,75,208,4,198,76,48,
     52,198,75
39 DATA 169,177,160,2,32,162,187,169,167,160,2,32,80,184,162,167,
     160,2
41 DATA 32,212,187,32,73,184,32,170,177,132,20,133,21,169,92,160,0
43 DATA 32,162,187,169,172,160,2,32,80,184,32,0,192,144,196
45 DATA 169,167,160,2,32,162,187,32,73,184,32,204,188,162,167,
     160,2,32,212,187
47 DATA 169,172,160,2,32,162,187,32,73,184,32,204,188,162,172,
     160,2,32,212,187
49 DATA 76,140,192
50 REM ----- FINE ROUTINE L.M. -----
51 REM
52 REM
59 REM ----- CARICA ROUTINE L.M. -----
60 FOR I=49152 TO 49585: READ P: POKE I,P:NEXT
69 REM ----- DEFINISCE CHIAMATE SUBR. E PULISCE SCHERMO GRAFICO ----
70 PL=49292: CL=49258: SYSCL
79 REM ----- INIZIALIZZA VIC II -----
80 VI=53248: M=VI+17: AD=VI+24: CI=56576
90 POKE M,PEEK(M) OR 32: POKE AD,7*16+8: POKE CI, PEEK(CI) AND 254
99 REM ----- SCELTA COLORI -----
100 POKE 2,207
109 REM ----- DISEGNA QUADRATI -----
110 SYSCL: FOR I=8 TO 80 STEP 8
120 SYSPL,160+I,100+I TO 160+I,100-I TO 160-I,100-I TO 160-I,100+I
     TO 160+I,100+I: NEXT
130 FOR I=0 TO 1000: NEXT
139 REM ----- DISEGNA FIGURE LISSAJUS -----
140 SYSCL: FOR I=0 TO 189: SYSPL TO 160+150*SIN(I/6),100+90*COS(I/10):
     NEXT
150 FOR I=0 TO 1000: NEXT
159 REM ----- RIPRISTINA SCHERMO TESTO -----
160 POKE M,PEEK(M) AND 223: POKE AD,21: POKE CI,PEEK(CI) OR 1
170 PRINT CHR$(147) TAB(207)"PER RIPETERE PREMI UN TASTO"
180 PRINT TAB(210)"PER USCIRE PREMI STOP"
190 GET K$: IF K$="" GOTO 190
200 RUN 70
```

Listato - Push Over

```
2 REM ++++++++++++++++++++++
3 REM PUSH OVER - PC CLUB N°5
4 REM GIOCO PER VIC 20
5 REM BY MARINAI SIMONE
6 REM ++++++++++++++++++++++
8 REM INIZIALIZZAZIONE SCHERMO
10 DIM A(4,4):G=1
20 POKE 36864,20 : POKE 36865,46 : POKE 36866,141 : POKE 36867,160
30 PRINT "(cls)(2 spazi)A()B()C()D()E(3 spazi)(176.96.178.96.178.96.178.96.
   178.96.174)()";
40 FOR P=70 TO 78 STEP 2
50 PRINT CHR$(P);"125.160.125.160.125.160.125.160.125.160.125)";CHR$(P+1);
60 PRINT "160.171.96.123.96.123.96.123.96.123.96.179.160)";
70 NEXT
80 PRINT "(13 Cs)(160.173.96.177.96.177.96.177.96.177.96.189)(3 spazi)P()Q
   ()R()S()T"
85 REM INPUT GIOCATORI
90 GOSUB 780:PRINT"GIOCATORE (•)";:INPUT G1$
100 GOSUB 780:PRINT"GIOCATORE (o)";:INPUT G2$
105 REM RICHIESTA E CONVALIDA MOSSA
110 IF G<>1 THEN140
120 GOSUB 780:PRINT G1$;" (•) COME MUOVI";: INPUT M$:M=ASC(M$):IF M<65 OR
   M>84 THEN120
130 S=G:G=0:GOTO160
140 GOSUB 780:PRINT G2$;" (o) COME MUOVI";: INPUT M$:M=ASC(M$):IF M<65 OR
   M>84 THEN140
150 S=G:G=1
155 REM SCELTA ROUTINES RELATIVE ALL'AGGIORNAMENTO DI MEMORIA
160 IF M<70 THEN 200
170 IF M>79 THEN 210
180 IF INT(M/2)=M/2 THEN 220
185 REM AGG. MEM. PER MOVIMENTO A SINISTRA
190 L=(M-69)/2-1:FORI=0 TO 3:A(I,L)=A(I+1,L) : NEXT : A(4,L)=S : C=L : D=C :
   E=0: F=4 : H=2 : GOTO 230
195 REM AGG. MEM. PER MOVIMENTO  IN BASSO
200 L=M-65 : FOR I=4 TO 1 STEP-1 : A(L,I)=A(L,I-1) : NEXT :A(L,0)=S : E=L :
   F=E : C=0 : D=4: H=1 : GOTO230
205 REM AGG. MEM. PER MOVIMENTO IN ALTO
210 L=M-80 : FOR I=0 TO 3 : A(L,I)=A(L,I+1) : NEXT :A(L,4)=S : E=L : F=E :
   C=0 : D=4 : H=1:GOTO 230
215 REM AGG. MEM. PER MOVIMENTO A DESTRA
220 L=(M-70)/2 : FOR I=4 TO 1 STEP-1 : A(I,L)=A(I-1,L): NEXT : A(0,L)=S:
   C=L : D=C : E=0 : F=4 : H=2
225 REM AGGIORNAMENTO SCHERMO E SOMMA DELLA LINEA RELATIVA ALLA MOSSA
230 T=0 : FOR B=C TO D
240 FOR A=E TO F
250 PRINT"(home)";SPC(28);SPC(2*A+2*B*13);
260 A1=A(A,B)
270 IF A1=0 THEN PRINT " ";: GOTO 300
280 IF A1=1 THEN PRINT "•";: GOTO 300
290 PRINT "o";
300 T=T+A1
310 NEXT : NEXT
320 A=A-1 : B=B-1 : GOSUB 550
345 REM SOMME DELLE LINEE PERPENDICOLARI ALLA MOSSA
350 T=0 : IF H=1 THEN 400
360 H=1 : FOR A=0 TO 4
370 FOR B=0 TO 4
380 T=T+A(A,B) : NEXT : B=B-1 : GOSUB 550
390 T=0 : NEXT : GOTO 450
400 H=2 : FOR B=0 TO 4
410 FOR A=0 TO 4
420 T=T+A(A,B) : NEXT: A=A-1 : GOSUB 550
430 T=0 : NEXT
445 REM SOMME DELLE LINEE DIAGONALI
450 T=0 : FOR A=0 TO 4 : T=T+A(A,A) : NEXT : H=3 : GOSUB 550
460 T=0 : FOR A=0 TO 3 : T=T+A(A,A+1) : NEXT : GOSUB 700
470 T=0 : FOR A=1 TO 4 : T=T+A(A,A-1)   NEXT : GOSUB 700
480 T=0 : FOR A=0 TO 4 : T=T+A(A,4-A)   NEXT : H=4 : GOSUB 550
490 T=0 : FOR A=0 TO 3 : T=T+A(A,3-A) : NEXT : GOSUB 700
500 T=0 : FOR A=1 TO 4 : T=T+A(A,5-A) : NEXT : GOSUB 700
510 GOTO 110
545 REM SUBROUTINE VERIFICA DELLA SOMMA
550 IF T>35 OR T=4 OR T=5 OR T=13 THEN 570
560 RETURN
570 ON H GOTO 580,600,620,640
580 IF A(A,1)=A(A,2) AND A(A,2)=A(A,3) THEN 750
590 RETURN
600 IF A(1,B)=A(2,B) AND A(2,B)=A(3,B) THEN 750
610 RETURN
620 IF A(1,1)=A(2,2) AND A(2,2)=A(3,3) THEN 750
630 RETURN
640 IF A(1,3)=A(2,2) AND A(2,2)=A(3,1) THEN 750
650 RETURN
695 REM VERIFICA SOMMA
700 IF T>35 OR T=4 OR T=5 OR T=13 THEN 750
710 RETURN
745 REM MESSAGGIO FINALE
750 IF T>35 THEN J$=G2$ : GOTO 770
760 J$=G1$
770 GOSUB 780 : PRINT J$;" HAI VINTO !!" : END
780 PRINT "(home)"; SPC(169);"(40 spazi)(home)";SPC(169);: RETURN
```

Utility e linguaggio macchina per C.64

Vi proponiamo due interessanti programmi, realizzati da **Massimo Roberto Garibaldi**, per il **Commodore 64**. Il primo è una Utility che serve a fare copie di blocchi di dischetti, sia che si possieda un solo drive sia che se ne possieda due. È possibile anche formattare i dischetti prima della copia, come è possibile anche fornire la traccia da cui si intende partire (vi ricordo che un dischetto contiene 35 traccie, e la traccia 1 è la più esterna). Il secondo programma inviato da Massimo è in linguaggio macchina e serve a leggere il valore delle due porte giochi presenti sul C-64 (potete avere Joystick, Light Pen, Paddle). Viene fornito il listato del programma BASIC che carica le routines in linguaggio macchina, oltre che al disassemblato completo e commentato delle routines.

Entrambi i listati sono fatti in una maniera particolare, nel senso che tutte le istruzioni che sarebbero su una riga separate dai due punti, sono invece scritte su più righe. Ricordatevi quindi, quando li digitate, di notare i numeri di riga, e mettere tutte le varie linee del listato che formano una riga insieme, separate dai soliti due punti.

L'unica grossa lacuna riguarda la documentazione (non ci sono REMarks, né liste delle variabili, né commenti ai listati). Penso di avervi annoiato abbastanza, per cui passo senz'altro la parola al nostro amico.

Copy Block Contents

Il Programma è particolarmente conciso **allo** scopo di lasciare molta memoria libera che viene usata come buffer.

Il Programma è infatti basato sulla gestione di una parte di memoria che viene caricata dal disco originale e scaricata su quello di backup.

Tabella controllori

A=USR(1)	LEGGE IL JOYSTICK DEL PORT NO.1
A=USR(2)	LEGGE IL JOYSTICK DEL PORT NO.2
A=USR(3)	LEGGE LA COORDINATA X DELLA LIGHT PEN
A=USR(4)	LEGGE LA LA COORDINATA Y DELLA LIGHT PEN
A=USR(5)	LEGGE LA PADDLE X SUL PORT 1
A=USR(6)	LEGGE LA PADDLE X SUL PORT 2
A=USR(7)	LEGGE LA PADDLE Y SUL PORT 1
A=USR(8)	LEGGE LA PADDLE Y SUL PORT 2

```
    5 PRINT"■■";
      FORP=1TO51
        PRINT" ";
      NEXT
      PRINT"GAME CONTROL READ ";
      FORP=1TO51
        PRINT" ";
      NEXT
      PRINT"■"
   10 PRINT"■■■ATTENDERE, PREGO■"
   20 E=53247
      FORX=E-160TOE
        READA
        POKEX,A
      NEXT
   30 S=E-160+8
      SH=INT(S/256)
      SL=S-SH*256
   40 POKE785,SL
      POKE786,SH
      SP$="      "
  100 PRINT"■■■JOYSTIK -A-                JOYSTIK -B-■"
      X=1
      GOSUB200
  110 PRINT"■■■LIGHT PEN -X-             LIGHT PEN -Y-■"
      X=3
      GOSUB200
  120 PRINT"■■■X PADDLE  -A-             X PADDLE  -B-■"
      X=5
      GOSUB200
  130 PRINT"■■■Y PADDLE  -A-             Y PADDLE  -B-■"
      X=7
      GOSUB200
  150 PRINT"■■■■■■"
      GOTO100
  200 Y=USR(X)
      Z=USR(X+1)
      A$=STR$(Y)
      B$=STR$(Z)
  210 A$=A$+LEFT$(SP$,6-LEN(A$))
      B$=B$+LEFT$(SP$,6-LEN(B$))
  220 PRINTTAB(6)A$TAB(30)B$
      RETURN
 5000 DATA167,162,182,187,219,212,205,198
 5010 DATA169,220,133,106,169,0,133,105
 5020 DATA169,212,133,108,169,25,133,107
 5030 DATA169,208,133,110,169,19,133,109
 5040 DATA169,207,133,252,133,254,169,94
 5050 DATA133,253,32,170,177,160,1,196
 5060 DATA101,240,8,192,8,240,11,200
 5070 DATA76,142,207,177,253,133,251,108
 5080 DATA251,0,96,160,0,76,169,207
 5090 DATA160,1,177,105,73,255,41,31
 5100 DATA168,169,0,32,145,179,96,160
 5110 DATA0,76,189,207,160,1,177,109
 5120 DATA168,169,0,32,145,179,96,160
 5130 DATA0,162,64,76,223,207,160,1
 5140 DATA162,64,76,223,207,160,0,162
 5150 DATA128,76,223,207,160,1,162,128
 5160 DATA128,173,2,220,133,2,169,192
 5170 DATA141,2,220,142,0,220,234,234
 5180 DATA234,234,177,107,168,169,0,32
 5190 DATA145,179,165,2,141,2,220,88
 5200 DATA96
```

Dopo, il RUN viene chiesto di quali unità si dispone. Se abbiamo un'unica unità, basterà confermare le risposte che sono presenti sullo schermo come default, se invece vogliamo che la copia avvenga tra diversi drives dovremo modificare i valori proposti e farli accettare con RETURN. Tutto questo permette l'utilizzo del programma anche su Dual Drive.
Verrà poi richiesto se utilizzare o meno la routine di formattazione. Prima che le operazioni di copia inizino dobbiamo ancora fornire al computer la traccia dalla quale vogliamo iniziare. Questo permette un notevole risparmio di tempo in quanto si copieranno solo le tracce che effettivamente contengono dei dati.
Inizia quindi la copia con le richieste di scambio tra il disco originale e di backup.

Il programma è completo di routine per il controllo di errori disco che vengono richiamate ad ogni suo accesso e di continui controlli sull'intervento dell'operatore le cui risposte sbagliate non vengono accettate. Tutto ciò porta ad una estrema facilità d'uso.
È da notare l'uso dell'istruzione WAIT alle linee **9**, **55** e **1010** dove il programma resta bloccato finché non viene premuto un carattere. La seguente GET A$, ripulisce la tastiera da caratteri che altrimenti sporcherebbero il video.

Game Read

Ho presentato il programma con due listati.
Il primo è in Basic e carica in memoria le routine in L.M. di lettura, l'altro è il disassemblato in mnemonico 6502 delle stesse.

Utility e linguaggio macchina

```
    0 PRINT"■■■______________________________"
      DIMAX(255,68)
      POKE53280,1
      POKE53281,1
    1 PRINT" COPY BLOCK CONTENTS"
    2 PRINT"‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾"
    3 INPUT"■■COPY FROM UNIT  8■■■";UF
      GOSUB800
      DF=VAL(D$)
    4 INPUT"COPY TO UNIT  8■■■";UT
      GOSUB800
      DT=VAL(D$)
    5 PRINT"■■FORMAT NEW DISK"
      PRINT"(Y/N) ? ";
    6 WAIT197,64,255
      GETA$
      PRINTA$
      IFA$="Y"THENGOSUB1000
    7 INPUT"■COPY FROM TRACK  18■■■■";T
      IFT<1ORT>35THENPRINT"■■";
                        GOTO7
    8 FORC=1TO10
        PRINT"■■■INSERT MASTER DISK"
        PRINT"AND PRESS A KEY"
    9   WAIT197,64,255
        GETA$
        PRINT"■■WAIT A MOMENT■"
   10   LF=15
        OPENLF,UF,15,"I"+STR$(DF)
        OPEN5,UF,5,"#"
        GOSUB300
        AA=T
        BB=S
        FORX=0TO68
   15     GOSUB500
   20     PRINT"                          ■"
          PRINT"TRACK"T"  SECTOR"S"■"
   25     PRINT#LF,"B-P:"5;0
          PRINT#LF,"U1:"5;DF;T;S
          GOSUB300
   30     FORL=0TO255
            GET#5,A$
            IFA$=""THENA=0
                        GOTO40
   35       A=ASC(A$)
   40       AX(L,X)=A
          NEXTL
        NEXTX
        CLOSELF
        CLOSE5
   50   PRINT"■■INSERT BACKUP DISK"
        PRINT"AND PRESS A KEY"
        T=AA
        S=BB
   55   WAIT197,64,255
        GETA$
        PRINT"■■WAIT A MOMENT■"
   60   LF=14
        OPENLF,UT,15,"I"+STR$(DT)
        OPEN5,UT,5,"#"
        GOSUB300
        FORX=0TO68
   70     PRINT#LF,"B-P:"5;0
          GOSUB300
   75     GOSUB500
          PRINT"                          ■"
          PRINT"TRACK"T"  SECTOR"S"■"
   80     FORL=0TO255
            PRINT#5,CHR$(AX(L,X));
          NEXTL
          PRINT#LF,"U2:"5;DT;T;S
        NEXTX
        GOSUB300
   90   CLOSELF
        CLOSE5
      NEXTC
      PRINT"■■COPY END"
      END
  300 REM ***********************
  305 REM **CHECK DISK ERROR**
  310 REM ***********************
  320 INPUT#LF,EN,EM$,ET,ES
      IFEN=0THENRETURN
  330 PRINT"■■■■DISK ERROR■"
      PRINTEN,EM$,ET,ES
      END
  500 REM ***********************
  501 REM **   ADG. T & S    **
  502 REM ***********************
  505 S=S+1
      IFT>0ANDT<18ANDS>20THENS=0
                                T=T+1
                                RETURN
  510 IFT>17ANDT<25ANDS>18THENS=0
                                T=T+1
                                RETURN
  520 IFT>24ANDT<31ANDS>17THENS=0
                                T=T+1
                                RETURN
  530 IFT>30ANDT<36ANDS>16THENC=10
                                GOTO90
  540 RETURN
  800 REM ***********************
  801 REM **  WICH DRIVE ?  **
  802 REM ***********************
  810 INPUT"DRIVE  0■■■";D
      IFD↑2<>DTHEN800
  820 D$=CHR$(D+48)
      RETURN
 1000 REM ***********************
 1001 REM **FORMAT  NEW DISK**
 1002 REM ***********************
 1005 INPUT"■■■DISK NAME";D$
      INPUT"■DISK ID";I$
 1010 PRINT"■■INSERT BACKUP DISK"
      PRINT"AND PRESS A KEY"
      WAIT197,64,255
      GETA$
 1020 PRINT"■WAIT A MOMENT"
 1030 OPEN15,UT,15,"N"+STR$(DT)+":"+D$+","+I$
      CLOSE15
      RETURN
```

Utility e linguaggio macchina

```
DIS ASSEMBLATO DELLE ROUTINES DI LETTURA
DELLE PORTE GIOCHI

CF5F:       A7          : Sono i dati che verranno usati per raggiungere
CF60:       A2          : le routine di lettura di Joystick, Paddle e Light Pen
CF61:       B6          :
CF62:       BB          :
CF63:       DB          :
CF64:       D4          :
CF65:       CD          :
CF66:       C6          :
CF67:  LDA DC           : Vettore in pagina zero ($69) per le locazioni $DC00-
CF69:  STA $6A          : $DC01 riguardanti il Complex Interface Adapter CIA#1
CF6B:  LDA #$00         :
CF6D:  STA $69          :
CF6F:  LDA #$D4         : Vettore in pagina zero ($6B) per le locazioni $D419-
CF71:  STA $6C          : $D41A riguardanti il Sound Interface Device
CF73:  LDA #$19         :
CF75:  STA $6B          :
CF77:  LDA #$D0         : Vettore in pagina zero ($6D) per le locazioni $D013-
CF79:  STA $6E          : $D014 riguardanti il Video Interface Controller VIC2
CF7B:  LDA #$13         :
CF7D:  STA $6D          :
CF7F:  LDA #$CF         : Vettore per i JMP indiretti
CF81:  STA $FC          : nella locazione $FC (parte alta dell'indirizzo)
CF83:  STA $FE          :
CF85:  LDA #$5E         : Vettore di partenza dati per i JMP indiretti $CF5E
CF87:  STA $FD          : nelle locazioni $FD e $FE (parte alta e parte bassa)
CF89:  JSR $B1AA        ; Il FPA#1 e' convertito in intero e messo in $64 e $6

CF8E:  CPY $65          ; Confronta Y con il contenuto di $65
CF90:  BEQ $CF9A        ; Se il cont.di $65=Y allora CF9A
CF92:  CPY #$08         ; Y ha raggiunto il 8 ?
CF94:  BEQ $CFA1        ; Se si allora esce
CF96:  INY              ; Continua il conteggio
CF97:  JMP $CF8E        ; E ritorna da capo
CF9A:  LDA ($FD),Y      ; Carica in A l'Yesimo dato a partire da $CF5E
CF9C:  STA $FB          ; Completa il vettore per il JMP indiretto
CF9E:  JMP ($00FB)      ; Salta alla locazione cosi' formatasi
CFA1:  RTS              ; Se il numero introdotto dall'USR > 8 allora ritorna
CFA2:  LDY #$00         ; Azzera il reg. Y
CFA4:  JMP $CFA9        ; Salta a $CFA9
CFA7:  LDY #$01         ; Carica il reg. Y con 1
CFA9:  LDA ($69),Y      ; Legge e mette in A il JOY il cui numero e' in Y
CFAB:  EOR #$FF         ; Simula una NOT logico con un OR esclusivo con 255
CFAD:  AND #$1F         ; Maschera i bit 5,6,7,8 del reg. A
CFAF:  TAY              ; Trasferisce A in Y
CFB0:  LDA #$00         ; Azzera A
CFB2:  JSR $B391        ; Trasforma A,Y in numero intero e lo mette nel FPA#1
CFB5:  RTS              ; Ritorna al BASIC
CFB6:  LDY #$00         ; Azzera il re. Y
CFB8:  JMP $CFBD        ; Salta a $CFBD
CFBB:  LDY #$01         ; Carica Y con 1
CFBD:  LDA ($6D),Y      ; Legge la Light Pen, la coordinata (x,y) dipende da Y
CFBF:  TAY              : Come dalla linea CFAF alla linea CFB2
CFC0:  LDA #$00         :
CFC2:  JSR $B391        :
CFC5:  RTS              ; Ritorna al BASIC
CFC6:  LDY #$00         : Definisce quale paddle e da quale port
CFC8:  LDX #$40         :
CFCA:  JMP $CFDF        ; Salta a $CFDF
CFCD:  LDY #$01         : Definisce quale paddle e da quale port
CFCF:  LDX #$40         :
CFD1:  JMP $CFDF        ; Salta a $CFDF
CFD4:  LDY #$00         : Definisce quale paddle e da quale port
CFD6:  LDX #$80         :
CFD8:  JMP $CFDF        ; Salta a $CFDF
CFDB:  LDY #$01         : Definisce quale paddle e da quale port
CFDD:  LDX #$80         :
CFDF:  SEI              ; Disabilita l'IRQ
CFE0:  LDA $DC02        : Il Data Direction Register e' depositato in $02
CFE3:  STA $02          :
CFE5:  LDA #$C0         : Predispone il DDR per l'input
CFE7:  STA $DC02        :
CFEA:  STX $DC00        ; Sceglie il paio di paddle definito da X
CFED:  NOP
CFEE:  NOP
CFEF:  NOP
CFF0:  NOP
CFF1:  LDA ($6B),Y      ; Legge la paddle definita da Y
CFF3:  TAY              : Come dalla linea CFAF alla linea CFB2
CFF4:  LDA #$00         :
CFF6:  JSR $B391        :
CFF9:  LDA $02          : Ripristina il DDR
CFFB:  STA $DC02        :
CFFE:  CLI              ; Riabilita l'IRQ
CFFF:  RTS              ; Ritorna al BASIC
```

Per realizzare il programma si hanno quindi due possibilità. Se si possiede un monitor che accetti la notazione standard MOS 6502 si può digitare il secondo listato, altrimenti si dovrà utilizzare il primo. Il programma in Basic si compone di tre parti.

La parte che carica in memoria le routine in L.M. (da 5 a 40), le linee di data che le rappresentano (da **5000** a **5200**) e un esempio di utilizzo del modulo in L.M. (da **100** a **220**). Quest'ultima, che in ogni modo non è indispensabile, può essere omessa, costituisce una maschera continuamente aggiornata contenente i valori del Joystick, delle Paddles e delle coordinate X ed Y della Light Pen. Vediamo ora come, una volta caricato il primo o il secondo programma, si utilizza il tutto.

L'uso di queste routine particolarmente veloci è estremamente semplice. Basterà infatti definire una variabile uguale alla funzione USR (x). In essa la x sarà un numero da 1 a 8 che corrisponderà ai controllori come da tabella.

La USR potrà anche essere posta a seguito di un'altra istruzione Basic in quanto essa stessa rappresenta il valore del controllore desiderato che in pratica viene ad essa sostituito.

Poiché per l'uso dello statment USR il computer ha bisogno di conoscere l'indirizzo di partenza della routine in L.M. esso si deve memorizzare in due bytes adiacenti (785 e 786) nella notazione Low end High Address; è quello che fanno le linee 30 e 40.

Nel disassemblato, comunque commentato, sono ancora da notare due particolarità.

L'utilizzo di una loop di riconoscimento del numero passato al L.M. attraverso la USR che costruisce un JMP indiretto alle routine necessarie. L'utilizzo della subroutine di sistema che trasforma il numero contenuto nell'Accumulatore 1 in virgola mobile in un numero intero e che lo memorizza in $64 z $65. La routine è richiamata in $B1AA e JSR $B1AA. La simulazione del NOT logico con un OR esclusivo con $FF alla linea $CFAB.

Aggiungo infine che i programmi Basic sono stati da me listati con un programma che evidenzia i loop e che divide in linee successive gli statment facenti parte di una stessa linea Basic. Si devono perciò battere tutti insieme e separati dal due punti «:».

Una «vita» per giocare

Vi proponiamo un'altra elaborazione del gioco «Vita», realizzata da **Antonio Morra** di Milano per i Computer Commodore.

Prima di presentare il programma apriamo una piccola parentesi per spiegare il meccanismo del gioco stesso. Inventato dal Matematico dell'università di Cambridge, John Horton, questo gioco, che ha come unico giocatore il computer, è stato riproposto da moltissimi altri matematici. Si gioca con una scacchiera illimitata, formata da innumerevoli caselle, dentro le quali nascono e muoiono un numero infinito di automi unicellulari. Data una certa configurazione iniziale, le regole con cui gli automi nascono, muoiono e si riproducono sono le seguenti: ogni automa ha nove caselle circostanti, se almeno due delle otto non contengono un automa il primo muore, mentre muore anche se i vicini sono più di tre. Ogni automa con vicino altri tre genera un nuovo essere nel turno successivo. Le morti e le nascite avvengono contemporaneamente.

La versione presentata dal nostro lettore è scritta completamente in Basic e può quindi girare tranquillamente con qualsiasi computer Commodore (ed anche di altre marche se si traducono le poche istruzioni differenti).

Il gioco si svolge in una griglia di 28 per 18 caselle. I frequenti REM aiutano a capire il programma.

Ecco la spiegazione delle principali routines:

1300-1380: Cerca gli indirizzi di partenza e di arrivo della parte piena della griglia.

1150-1240: Indicizza le cellule vuote immediatamente vicine a quelle piene.

1000-1130: Routine per la stampa del tempo del gioco e dello «STOP» dopo ogni generazione. La continuazione avviene dopo aver premuto la barra di spazio oppure automaticamente dopo 10 secondi senza interventi.

10000-: Programma vero e proprio con formazione della griglia, input iniziali ed elaborazioni varie.

Il programma è stato costruito con un Commodore 64 ma, come già detto, è sufficiente eliminare i simboli del colore (righe 110, 220 e 10007) perché giri perfettamente su qualsiasi Commodore.

Listato - Vita per giocare

```
100 PRINT"☐■":DIMVI%(18,28):IR%=18:IC%=28
110 CO$(1)="◣☒ ■■":CO$(2)="+":CO$(3)="*":IR%=18:IC%=28
120 SP$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
130 PRINT"▒▒▒▒      *****************************     "
140 PRINT"      *          V I T A          *   "
150 PRINT"      *"TAB(33)"*"
160 PRINT"      *             BY            *   "
170 PRINT"      *"TAB(33)"*"
180 PRINT"      *        A. MORRA  MILANO   *   "
190 PRINT"      *"TAB(33)"*"
200 PRINT"      *****************************    ▒▒▒▒"
210 PRINT"      (▒PREMI UN TASTO PER INIZIARE■);☐"
220 GETA$:IFA$=""THEN220
230 PRINT"☐":GOTO10000
240 :
250 :
260 :
1000 REM
1050 PRINT"▒▒▒▒"SPC(32)TIME$:RETURN
1060 :
1080 FG$="":PRINTSP$"▒▒     'SPACE' PER CONTINUARE        "
1090 PRINT"▒▒▒▒▒▒"TAB(33)"STOP":T%=VAL(RIGHT$(TIME$,1))-1:IFT%<0THENT%=9
1100 GETFG$:IFFG$=""ANDVAL(RIGHT$(TIME$,1))<>T%THEN1100
1110 PRINT"▒▒▒▒▒▒"TAB(33)"     "
1120 PRINTSP$"▒▒                              "
1130 GOSUB1050:FG$="":RETURN
1140 :
1150 IFRW=1THEN1190:REM **** CREA I VICINI VUOTI **********
1160 VI=VI%(RW-1,CW):IFVI=0THENVI%(RW-1,CW)=4:REM              AC ←←←←←←←←←←
1170 IFCW>1THENVI=VI%(RW-1,CW-1):IFVI=0THENVI%(RW-1,CW-1)=4:REM AS ←←←←←←←←←←
1180 IFCW<28THENVI=VI%(RW-1,CW+1):IFVI=0THENVI%(RW-1,CW+1)=4:REM AD ←←←←←←←←←←
1190 IFCW>1THENVI=VI%(RW,CW-1):IFVI=0THENVI%(RW,CW-1)=4:REM     FS ←←←←←←←←←←
1195 IFCW<28THENVI=VI%(RW,CW+1):IFVI=0THENVI%(RW,CW+1)=4:REM    FD ←←←←←←←←←←
1200 IFRW=18THENRETURN
1210 IFCW>1THENVI=VI%(RW+1,CW-1):IFVI=0THENVI%(RW+1,CW-1)=4:REM BS ←←←←←←←←←←
1220 VI=VI%(RW+1,CW):IFVI=0THENVI%(RW+1,CW)=4:REM              BC ←←←←←←←←←←
1230 IFCW<28THENVI=VI%(RW+1,CW+1):IFVI=0THENVI%(RW+1,CW+1)=4:REM BD ←←←←←←←←←←
1240 RETURN
1250 :
1260 :
1270 :
1280 :
1290 :
1300 IFRW<=IR%THENIR%=RW-1:REM INIZIO RI
1310 IFCW<=IC%THENIC%=CW-1:REM INIZIO CO
1320 IFRW>=FR%THENFR%=RW+1:REM FINE RIGA
1330 IFCW>=FC%THENFC%=CW+1:REM FINE COLO
1340 IFIR%<1THENIR%=1
1350 IFIC%<1THENIC%=1
1360 IFFR%>18THENFR%=18
1370 IFFC%>28THENFR%=28
1380 RETURN
1390 :
1400 :
1410 :
1420 :
1430 :
10000 PRINT"☐                 1111111111122222222"
10007 PRINT"    1234567890123456789012345678 ☒■TEMPO■☐"
10010 FORRW=1TO18:PRINTRW;:IFRW>9THENPRINT"▮";
10007 PRINT"    1234567890123456789012345678 ☒■TEMPO■☐"
10010 FORRW=1TO18:PRINTRW;:IFRW>9THENPRINT"▮";
10020 FORCW=1TO28:PRINT"┌";:NEXT:PRINT"| ":NEXT:PRINT"▮■▮▮";:FORW=1TO28:PRINT"¯"
NEXT
10030 PRINT:PRINT"■INSERIRE I PRIMI ORGANISMI UNICELLULARI"
10040 INPUT"QUANTE CELLULE INIZIALI ";A$
10050 IFVAL(A$)<3ORVAL(A$)>15THENPRINT"☐";:GOTO10040
```

Un «Renumber» per il Vic 20

Abbiamo scelto questo programma di «Renumber», realizzato da **Giancarlo Ammirata**, perché pensiamo possa interessare un gran numero di lettori proprietari di un Commodore VIC 20.

Le linee di qualsiasi programma vengono memorizzate dal VIC a partire dalla prima locazione di memoria disponibile, sotto forma di numeri. Il primo numero è sempre uno 0 che ha lo scopo di comunicare al sistema operativo l'inizio di una nuova linea di programma. I successivi due numeri sono i byte di LINK. Moltiplicando il valore del secondo per 256 ed aggiungendo il valore del primo si ottiene l'indirizzo di memoria della successiva linea di programma, numero che si ottiene sempre moltiplicando il secondo byte per 256 ed aggiungendo il primo.

A questo punto iniziano le vere e proprie istruzioni del programma. Ogni lettera, numero, istruzione BASIC, segno di punteggiatura, ecc. viene tradotta dal VIC in un numero che corrisponde ad esso, risparmiando così una buona quantità di memoria. Vediamo, ad esempio, come viene immagazzinata in memoria la linea:

100 A=A+1:IFA>256 THENA=0

Supponiamo di non avere espansioni di memoria. La prima locazione di memoria disponibile sarà allora la 4096:

4906:0

4097:22 4097:16 > Byte di LINK – 16*256+12=4118 (indirizzo d'inizio della successiva riga di progr.)

4099:100 4100:0 > Numero di linea

4101:65 > lettera A

4102:178 > segno =

4103:65 > lettera A

4104:170 > segno +

4105:49 > valore 1
4106:58 > segno :
eccetera.

Alterando quindi le opportune locazioni di memoria è possibile cambiare i valori dei vari numeri di linea. Il programma da renumerare deve necessariamente avere i numeri di linea seguenti le istruzioni del tipo GOTO, THEN, GOSUB, ecc. composti da cinque cifre. Così il numero 100 deve diventare 00100.

Il problema sorge dal fatto che se il nuovo numero è più lungo di quello già esistente bisognerebbe spostare verso il basso tutte le locazioni di memoria che seguono per far posto al (o ai) caratteri in più. Suggerisco, eventualmente, di usare una piccola routine in linguaggio macchina, poiché il basic è troppo lento.

Vediamo ora come opera in dettaglio il programma: linee

60006-60050
Conta il numero di linee del programma da renumerare

60060-60090
Crea il vettore A () con i vecchi numeri di linea

60100-60280
Controlla i rimanenti numeri di linea seguenti le istruzioni tipo GOTO, GOSUB THEN

60290-60310
Incrementa del passo il nuovo valore del numero di linea. Il programma gira con il VIC 20 senza espansioni. In caso contrario basta variare il valore della variabile P alle linee 60000, 60060, 60100 con quello d'inizio della memoria libera con la relativa espansione (1024 con la 3K, 4608 con la 8 o la 16K).

Per variare il passo di renumerazione basta variare PAS alla linea 60100. Ecco l'elenco delle variabili usate:

P = inizio memoria libera
C = Numero di righe di cui è formato il programma da renumerare
BL = Byte di LINK
NL = Numero di linea considerato
A () = Vettore con i vecchi numeri di linea
AL = Byte alto del numero di riga
BA = Byte basso del numero di riga
IN = Numero di riga di partenza
PAS = Passo d'incremento
CO = lunghezza del numero da cambiare
V$ = Stringa contenente il valore da cambiare
V = Valore da cambiare
MN = Posto nel vett. A () dove si trova il valore da cambiare
NV = Valore nuovo da sostituire

Questo programma va digitato in coda al programma da renumerare, il quale deve avere i numeri di riga più bassi di 60000.

Per farlo girare basta dare un RUN60000.

Listato - Vita per giocare

```
10060 PRINT"]]                                          ":VO%=VAL(A$)
10066 PRINT"                                          ]]":PRINT"]"
10070 FORW=1TOVO%
10080 PRINTW"* CELLULA "
10090 INPUT"■RIGA■ ";A$
10095 IFVAL(A$)<1ORVAL(A$)>18THENPRINT"]";:GOTO10090
10100 PRINT"]";
10110 INPUT"■COLONNA■ ";B$
10115 IFVAL(B$)<1ORVAL(B$)>28THENPRINT"]";:GOTO10110
10120 PRINT"]                    ":PRINT"]]";
10130 RW=VAL(A$):CW=VAL(B$):IF VI%(RW,CW)=1THENPRINT"■";:GOTO10090
10140 VI%(RW,CW)=1:GOSUB1150:GOSUB1300
10150 PRINT"■■■";:PRINTSPC(CW);:FORWW=1TORW:PRINT"■";
10160 NEXT:PRINT"■ ■"
10170 PRINTSP$:NEXT
10180 PRINTSP$"■     ***** 1" GENERAZIONE *****                  ":NG=1
10190 TIME$="000000":GOSUB1050
10200 :
10210 :
10220 :
10230 :
10240 :
10500 REM ***SCANNING RETICOLO*********
10510 FG$="":FORRW=IR%TOFR%:FORCW=IC%TOFC%:IFVI%(RW,CW)=0THEN12000
10600 TV%=0:IFRW=1THEN10640
10610 IFCW<15THENVI=VI%(RW-1,CW+1):IFVI=1ORVI=2THENTV%=TV%+1:REM AD <<<<<<<
10620 VI=VI%(RW-1,CW):IFVI=1ORVI=2THENTV%=TV%+1:REM             AC <<<<<<<
10630 IFCW>1THENVI=VI%(RW-1,CW-1):IFVI=1ORVI=2THENTV%=TV%+1:REM AS <<<<<<<
10640 IFCW>1THENVI=VI%(RW,CW-1):IFVI=1ORVI=2THENTV%=TV%+1:REM   FS <<<<<<<
10650 IFCW<28THENVI=VI%(RW,CW+1):IFVI=1ORVI=2THENTV%=TV%+1:REM  FD <<<<<<<
10655 IFTV%>3THEN10700
10660 IFRW=18THEN10700
10670 IFCW>1THENVI=VI%(RW+1,CW-1):IFVI=1ORVI=2THENTV%=TV%+1:REM BS <<<<<<<
10680 VI=VI%(RW+1,CW):IFVI=1ORVI=2THENTV%=TV%+1:REM             BC <<<<<<<
10690 IFCW<28THENVI=VI%(RW+1,CW+1):IFVI=1ORVI=2THENTV%=TV%+1:REM BD <<<<<<<
10700 REM
10710 VI=VI%(RW,CW)
10720 REM  SE MUORE
10730 IFVI=4THEN10770
10740 IFTV%<2ORTV%>3THENVI%(RW,CW)=2:FG=1:GOSUB1300
10750 REM  SE GENERA
10760 IFVI<>4THEN12000
10770 IFTV%=3THENVI%(RW,CW)=3:FG=1:GOSUB1150:GOSUB1300
10780 :
10820 :
12000 NEXT:GOSUB1050:NEXT:GOSUB1050
12010 REM QUADRO MUTAZIONI
12020 FORRW=IR%TOFR%:FORCW=IC%TOFC%
12030 IFVI%(RW,CW)=0THEN12060
12040 PRINT"■■■";:PRINTSPC(CW);:FORW=1TORW:PRINT"■";:NEXT
12050 PRINTCO$(VI%(RW,CW))
12060 NEXT:GOSUB1050:NEXT:GOSUB1050:GOSUB1080
12080 :
12090 :
12100 REM QUADRO NUOVA GENERAZIONE
12110 FORRW=IR%TOFR%:FORCW=IC%TOFC%
12120 IFVI%(RW,CW)=0THEN12160
12130 PRINT"■■■";:PRINTSPC(CW);:FORW=1TORW:PRINT"■";:NEXT
12140 IFVI%(RW,CW)=1ORVI%(RW,CW)=3THENPRINTCO$(1):VI%(RW,CW)=1
12150 IFVI%(RW,CW)=2THENPRINT" ":VI%(RW,CW)=4
12160 NEXT:NEXT:GOSUB1050:NG=NG+1
12170 PRINTSP$"■     *****"NG
12180 IFFGTHENFG=0:GOTO10500
12190 PRINTSP$"■     ■FINE EVOLUZIONE■                   ";:END

READY.
```

Listato - Renumber

```
60000 REM********************
60001 REM*     RENUMBER      *
60002 REM*        BY         *
60003 REM*GIANCARLO AMMIRATA*
60004 REM*  PER P. C. CLUB   *
60005 REM********************
60006 P=4096:C=-1
60010 DEFFNBL(X)=PEEK(X+2)*256+PEEK(X+1)
60020 DEFFNNL(X)=PEEK(X+4)*256+PEEK(X+3)
60030 BL=FN BL(P)-1:NL=FNNL(P)
60040 IFNL>60000THEN60060
60050 P=BL:C=C+1:GOTO60030
60060 DIMA(C-1):P=4096
60070 FORI=0TOC-1
60080 BL=FNBL(P)-1:NL=FNNL(P)
60090 A(I)=NL:P=BL:NEXTI
60100 P=4096:AL=00:BA=10:IN=BA:PAS=10
60110 FORT=1TOC:BL=FNBL(P)-1
60120 POKEP+4,AL:POKEP+3,BA
60130 FORJ=PTOBL:CO=0:V$=""
60140 IFPEEK(J)<>137ANDPEEK(J)<>141ANDPEEK(J)<>167THEN60280
60150 IFPEEK(J+1)<48ORPEEK(J+1)>57THEN60280
60160 FORKK=1TO5
60170 V$=V$+STR$(PEEK(J+1)-48):J=J+1:V=VAL(V$):CO=CO+1
60180 IFPEEK(J+1)<48ORPEEK(J+1)>57THENMN=1:GOTO60200
60190 NEXTKK
60200 FORK=0TOC-1
60210 IFA(K)=VTHEN60230
60220 MN=MN+1:NEXTK
60230 NV=IN+PAS*(MN-1):NV$=STR$(NV)
60240 FORL=2TOLEN(NV$)
60250 POKEJ-CO+L-1,VAL(MID$(NV$,L,1))+48
60260 NEXTL
60265 IFLEN(NV$)=5THEN60280
60270 FORL=LEN(NV$)TO5:POKEJ-CO+L,32:NEXT
60280 NEXTJ
60290 P=BL:BA=BA+PAS
60300 IFBA>255THENBA=BA-256:AL=AL+1:GOTO60300
60310 NEXTT

READY.
```

Listato - Allunaggio

```
1Ø  REM ALLUNAGGIO
2Ø  REM BY LUCA ARCOSTANZO PER PC CLUB
3Ø  REM ISTRUZIONI
4Ø  PRINT"CLR/HOME"
5Ø  PRINT"DEVI GUIDARE L'ALLUNAGGIO DI UNA NAVICELLA"
6Ø  PRINT"CHE SI TROVA A 120 KM D'ALTEZZA SULLA LUNA"
7Ø  PRINT"CON VELOCITA' INIZIALE DI 3600 KM/H;HAI A "
8Ø  PRINT"DISPOSIZIONE,OGNI 10 SECONDI, DA 8 A 200  "
9Ø  PRINT"KG DI CARBURANTE PER LA FRENATA (PUOI AN- "
1ØØ PRINT"CHE RINUNCIARVI E PROSEGUIRE IN CADUTA LI-"
11Ø PRINT"BERA);LA SCORTA INIZIALE E' DI 1600 KG.   "
12Ø PRINT"BUONA FORTUNA! PREMI UN TASTO PER CONTINUARE"
13Ø GET A$: IF A$=" " THEN 13Ø
14Ø PRINT"CLR/HOME"
15Ø PRINTTAB(5)"SEC";TAB(1Ø)"ALT";TAB(15)"VEL";TAB(2Ø)"CARB";TAB(25)"SERB"
16Ø PRINT
17Ø REM INIZIALIZZAZIONE CONDIZIONI INIZIALI
18Ø CR=16ØØ:TE=Ø:VØ=1:SØ=12Ø
19Ø INPUT"KG";KG
2ØØ IF(KG<>Ø)AND((KG<8)OR(KG>2ØØ))THEN19Ø
21Ø AC=Ø.ØØ16-(KG/5ØØØ)
22Ø CR=CR-KG
23Ø IF CR<Ø THEN  33Ø : REM CONTROLLO LIVELLO SERBATOIO
24Ø FOR I=Ø TO 9.9 STEP Ø.1 : REM INIZIO DEL CICLO DI DIECI SECONDI
25Ø VE=VØ+AC*I : REM CALCOLO VELOCITA'
26Ø AL=12Ø-(VE*I+AC*(I*I)/2)/1ØØØ : REM CALCOLO ALTEZZA
27Ø IF AL < =Ø THEN 35Ø  : REM CONTROLLO ALTIMETRO
28Ø PRINT"(2CURSOR UP)"
29Ø PRINTTAB(5)INT(I*1Ø)/1Ø;TAB(1Ø)INT(AL*1ØØ)/1ØØ;TAB(15)INT(VE*1ØØ)*36;
    TAB(2Ø)INT(KG);TAB(25)CR
3ØØ NEXT I
31Ø TE=TE+1Ø:SØ=AL:VØ=VE  :  REM AGGIORNAMENTO VARIABILI
32Ø GOTO 19Ø
33Ø PRINT"HAI FINITO IL CARBURANTE A DISPOSIZIONE !"
34Ø PRINT"SEIAA";AL;"KM DI ALTEZZA... ADDIO... ":GOTO 4ØØ
35Ø IF VE 1Ø THEN 38Ø
36Ø PRINT"SEI ALLUNATO DOPO";TE+I;"SECONDI ALLA VELOCITA'"
37Ø PRINT"DI";VE;"KM/H! LA NAVICELLA E' DISTRUTTA!":GOTO 4ØØ
38Ø PRINT"SEI FELICEMENTE ALLUNATO DOPO";TE+I;"SECONDI"
39Ø PRINT"HAI CONSERVATO";CR;"KG DI CARBURANTE;COMPLIMENTI!"
4ØØ PRINT"VUOI CONTINUARE?(S/N)"
41Ø GET A$: IF A$=" " THEN41Ø
42Ø IF A$="S"THEN14Ø
43Ø STOP
```

Allunaggio con il C.64

«Allunaggio» è uno dei più classici giochetti per calcolatrici tascabili, Home e Personal Computer: alla guida di un modulo lunare dovete scegliere la strategia ottimale di uso dei retrorazzi per frenare progressivamente senza schiantarvi al suolo in preda alla gravità del pianeta.

Il listing che proponiamo, scritto per CBM 64 dall'amico torinese **Luca Arcostanzo,** uno dei più fedeli sostenitori della rubrica giochi, ha il pregio di essere molto chiaro sia per il programmatore sia per chi gioca, e l'adattamento a un eventuale altro hardware dovrebbe risultare molto facile. I dati del vostro computer di bordo vengono aggiornati in tempo reale: tenetevi forte e andate a cominciare!

Sprite-Editor grafico per Commodore 64

Prima di passare alla descrizione del listato di **Enzo Farina** di Grugliasco facciamo una piccola introduzione tratta dall'articolo a cui si riferisce il lettore cioè «Uno Sprite Editor per gestire la grafica» apparso sul n. 3 del 15 settembre 1983 di Personal Computer.

Per prima cosa ricordiamo che uno Sprite è un'entità ad alta risoluzione costituita da 21 righe di 24 punti ciascuna. Ogni Sprite (fino ad un massimo di otto contemporaneamente) è gestito dall'integrato che si cura della gestione video (Video Interface Chip). Per programmarne uno bisogna definire la sequenza di byte, selezionare il colore e scegliere la posizione. Cambiando i valori di **X** e **Y** è possibile muovere lo Sprite.

Il programma presentato permetteva di costruire velocemente uno Sprite, visualizzandolo mentre lo si sta creando, correggerlo e memorizzarlo su nastro. Per ulteriori chiarimenti consigliamo comunque di leggere l'articolo in questione.

Passiamo ora al programma inviatoci dal nostro lettore. Le modifiche consigliate al programma originale consentono la programmazione degli Sprite anche in alta risoluzione e non solo in multicolor.

L'opzione viene richiesta al-

Listato - Sprite Editor

```
20 V=53248:POKE V+21,0:POKE V+23,0:POKE V+29,0:POKE V+28,0:RESTORE
21 FOR X=0TO15:READ A$(X):NEXT
30 FOR X=1TO21:FORC=1TO24:A(R,C)=46:NEXT:NEXT:FOR X=1TO63:B(X)=0:NEXT
50 FOR X=1TO63:POKE 831+X,B(X):NEXT:POKE V+21,4
51 PRINT "(clr)(cg)(rev) MODE(cg)":PRINT "M MULTICOLOR"
52 PRINT "H HI-RES"
53 GET A$:IF A$="" THEN 53
54 IF A$="M" THEN POKE V+28,4:CM=2:GOTO 60
55 IF A$="H" THEN POKE V+28,0:CM=1:GOTO 60
56 GOTO 53
70 PRINT"<= CAN.":IF CM=1 THEN PRINT "2 SC - BLU CH.":GOTO 90
160 R=S+X+(Y-1)*40:C=A(Y,X):POKE R,C:IF CM=2 THEN POKE R+1,C
180 IF A$="(cd)" THEN X=X+CM:IF X>24 THEN X=1
185 IF A$="(cs)" THEN X=X-CM:IF X<1 THEN X=23
190 IF A$="<=" THEN A(Y,X)=46:IF CM=2 THEN A(Y,X+1)=46
195 IF A$>"0" AND A$<"4" AND CM=2 THEN NR=48+VAL(A$):A(Y,X)=R:
    A(Y,X+1)=R
196 IF A$="2" AND CM=1 THEN A(Y,X)=50
290 R=S+X+(Y-1)*40:C=(A(Y,X)+128):POKE R,C:IF CM=2 THEN POKE R+1,C
291 GOTO 150
300 IF CM=2 THEN 312
302 Y=0:FOR R=1 TO 21:FOR X=0 TO 2:Y=Y+1:B(Y)=0:FOR C=0 TO 7:
    IF A(R,X*8+C+1)<>50 THEN 306
304 B(Y)=B(Y) OR 2 (7-C)
306 NEXT:NEXT:NEXT:FOR X=1 TO 63:POKE 831+X,B(X):NEXT:GOTO 140
312 come la linea del programma originale
314   "    "    "    "      "        "
316   "    "    "    "      "        "
310  cancellare
320    "
330 PRINT "(home)(16 cg)(rev)"A$": SI O NO"
410 POKE V+R,C:PRINT"(home)   (3 cg)";:IF CM=2 THEN 418
411 IF R=33 THEN 450
412 IF R=41 THEN 440
413 RETURN
418 IF R=33 THEN 450
```

l'inizio, e non può essere modificata durante l'Editing: il disegno si ottiene digitando il «2» (1 e 3 sono inattivi e non compaiono del resto neanche nel menu); per il resto l'uso è invariato. Pubblichiamo nel riguardo l'elenco delle linee da modificare o da inserire ex novo o da cancellare.

Leggere i diversi floppy

Nella gamma Commodore vi sono attualmente più unità a doppio floppy:

CBM 3040, DOS 1.0
CBM 4040, DOS 2,1
CBM 8050, DOS 2,5

E vi sono dei problemi d'incompatibilità fra questi differenti supporti. In certe applicazioni può tornar utile disporre di una procedura che riconosca ciascun tipo di dischetto evitando quindi possibili errori. Come proce-

Listato - Leggere i floppy

```
            LISTATO • **  * **
10  REM DEFINIZIONE DELLA UNITA' DISCHETTI
20  REM
100 OPEN80,8,15,"I0"
120 OPEN87,8,7,"0$"
130 GET#87,A$,CLOSE87
140 IF A$=CHR$(1) THEN PRINT "CBM 3040":END
150 IF A$="A" THEN PRINT "CBM 4040 ":END
160 PRINT "CBM 8050":END
```

FIGURA 1

	CBM 3040/4040	CBM 8050
NOME DEL DISCHETTO	143 - 159	5 - 20
ID	161.162	323.24

LISTATO 2

```
*.
10 REM LETTURA DEL NOME O DELLA - ID - DI UN DISCHETTO
100 N=5:REM PER CBM 8050 ALTIMENTI N=143
110 OPEN80,8,15,"I0"
120 OPEN86,8,6,"$0"
130 FOR I=1 RO N-1:GET#86,A$:NEXT I
140 N$="":FOR I=1 TO 16
150 GET#86,A$:IF A$=CHR$(160) THEN 170
160 N$=N$+A$
170 NEXT I
180 GET#86,A$,A$
190 I$="":FOR I=1 TO 2
200 GET#86,A$:IF A$=CHR$(160) THEN220
210 I$=I$+A$
220 NEXT I
230 PRINT "NOME  = ",N$," ID  = ",I$
240  END
```

Listato - Caccia di stelle

```
1 PRINT"J":POKE36879,250:POKE36869,240
2 PRINT" STAR CATCHER "
3 PRINT" Z=LT X=UP  .=DN /=RT "
4 PRINT"AFTER  8  STARS  THE  2ND SPACESHIP STARTS"
5 PRINT" KEEP AWAY FROM MINES"
6 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT
7 DATA24,24,60,60,60,126,102,102,102,102,126,60,60,60,24,24
8 DATA0,7,63,252,252,63,7,0,0,224,252,63,63,252,224,0
9 DATA24,36,231,102,60,60,66,129,24,24,24,231,231,24,24,24
10 DATA0,0,0,24,24,0,0,0,153,66,36,129,129,36,66,153,60,126,255,129,66,36,24,60
11 FORR=7432TO7432+71:READA:POKER,A:NEXTR
12 PRINT"J":POKE36869,255:POKE36879,25
13 V1=36878:S1=36874:POKEV1,15:S4=36877:Z=30720
14 FORD=1TO25
15 H=INT(RND(1)*350)+7768
16 POKEH,38:POKEH+Z,6:NEXTD
17 X=10:Y=7856:POKEX+Y,33:POKEX+Y+Z,2
18 P=10:F=8076:POKEP+F,37:POKEP+F+Z,4
19 A=10:B=8120:POKEA+B,37:POKEA+B+Z,4
20 TI$="000000":T=0:SS=33:S=0
23 PRINT"        -FLIGHT-TIME"
24 PRINT" *  STARS HIT        "
28 GOSUB30:GOSUB80:GOSUB30:GOSUB100:GOSUB30:GOTO28
30 REM*MOVE ROCKET*
31 PRINT" *"TI$:GETA$
32 IFA$="Z"THENS=1:SS=35
34 IFA$="/"THENS=2:SS=36
36 IFA$="X"THENS=3:SS=33
38 IFA$="."THENS=4:SS=34
39 POKES1,240
40 IFS=1THENPOKEX+Y,96:POKEX+Y+Z,1:X=X-1
42 IFS=2THENPOKEX+Y,96:POKEX+Y+Z,1:X=X+1
44 IFS=3THENPOKEX+Y,96:POKEX+Y+Z,1:Y=Y-22
46 IFS=4THENPOKEX+Y,96:POKEX+Y+Z,1:Y=Y+22
50 IFY=<7746THENS=4:SS=34:Y=Y+22
52 IFX=22THENS=1:SS=35:X=X-1
54 IFY=>8164THENS=3:SS=33:Y=Y-22
56 IFX=-1THENS=2:SS=36:X=X+1
57 IFPEEK(X+Y)=PEEK(F+P)THENGOTO120
58 IFPEEK(X+Y)=41THENGOTO120
59 IFPEEK(X+Y)=38THENGOSUB150
61 POKES1,0
62 POKEX+Y,SS:POKEX+Y+Z,2:RETURN
80 REM*MOVE 1ST SHIP*
81 BC=1:BL=32:BO=INT(RND(1)*16)
82 IFBO=1THENBL=41:BC=5:POKES1,230
83 IFPEEK(X+Y)=PEEK(F+P)THENPOKES1,0:GOTO120
84 IFY<FTHENF=F-22:POKEF+P+22,BL:POKEF+P+22+Z,BC:POKEF+P,37:POKEF+P+Z,4
86 IFY>FTHENF=F+22:POKEF+P-22,32:POKEF+P,37:POKEF+P+Z,4
88 IFX<PTHENP=P-1:POKEF+P+1,32:POKEF+P,37:POKEF+P+Z,4
90 IFX>PTHENP=P+1:POKEF+P-1,BL:POKEF+P+Z-1,BC:POKEF+P,37:POKEF+P+Z,4
96 POKES1,0
98 RETURN
100 REM*MOVE 2ND SHIP*
101 IFT=<8THENRETURN
102 BC=1:BL=32:BO=INT(RND(1)*4)
104 IFBO=2THENBL=41:BC=5:POKES1,220
108 IFY<BTHENB=B-22:POKEA+B+22,BL:POKEA+B+22+Z,BC:POKEA+B,37:POKEA+B+Z,4
110 IFY=>BTHENB=B+22:POKEA+B-22,BL:POKEA+B-22+Z,BC:POKEA+B,37:POKEA+B+Z,4
112 IFX<ATHENA=A-1:POKEA+B+1,BL:POKEA+B+1+Z,BC:POKEA+B,37:POKEA+B+Z,4
114 IFX>ATHENA=A+1:POKEA+B-1,BL:POKEA+B+Z+-1,BC:POKEA+B,37:POKEA+B+Z,4
116 POKES1,0:RETURN
120 REM*CRASH*
121 POKES4,220:POKEX+Y,39:POKEX+Y+Z,2
122 FORL=15TO0STEP-1:POKEV1,L
124 FORM=1TO200:NEXTM
126 POKEX+Y,40:POKEX+Y+Z,2:NEXTL:POKES4,0
128 PRINT"GAME   OVER"
130 PRINT"  PLAY AGAIN Y OR N"
132 GETA$:IFA$="Y"THENRUN
134 IFA$="N"THENPRINT"J":GOTO160
136 GOTO132
150 REM*HIT STAR*
151 POKES1,250
152 T=T+1
153 PRINT""T
154 POKES1,0
156 RETURN
160 PRINT" *PLEASE TURN ME OFF*"
162 GOTO162
170 REM YOU MUST CATCH THE STARS.
171 REM AFTER 8 STARS THE 2ND SHIP WILL START.
172 REM THEY WILL LAY MINES KEEP AWAY.
173 REM KEEP GOING.
```

dere se un programma lavora su questa unità floppy o su quest'altra? Leggendo la **Directory ($0 o $1)**. Nel primo carattere letto si determinerà su quale unità è stato formattato il dischetto:
CBM 3040 CODE ASCII PRIMO CARATTERE = 1
CBM 4040 CODE ASCII PRIMO CARATTERE = 65(=«A»)
CBM 8050 CODE ASCII PRIMO CARATTERE = 67(=«C»)
Il listato 1 vi dimostra quanto asserito.
Questo metodo non determina direttamente su quale unità di dischetto si sta lavorando, ma su quale supporto è stato formattato. Attenzione però, è pericoloso scrivere su un dischetto 4040 con un 3040. Il metodo che vi presentiamo è particolarmente utile in questo caso. Quando avviene la formattazione si fornisce al sistema un nome (fino a sedici caratteri) ed un identificatore *ID* di due caratteri.
Quando il nome è formato da meno di 16 caratteri esso viene completato da **«Shift Space», code Ascii: 160.**
La stessa cosa è valida per la - ID - per questo motivo si sconsiglia di mettere alcuni «Shift Space» nel nome o nella ID di un dischetto. Per leggere la ID faremo uso del programma con listato 2, nulla ci vieta di combinare il programma di listato uno con quello di listato 2.
In queste procedure viene utilizzato il Drive 0, per il Drive 1 bisogna sostituire I0 e $0 con J1 e $1.

A caccia di stelle sul VIC 20

Un bel programma di conquista stellare, abbastanza corto ma ricco di interesse, va segnalato senz'altro ai lettori anche se le istruzioni del listato sono in lingua inglese.
L'obiettivo del gioco è di andare a zonzo per un universo popolato di stelle, demolendo tutto quello che capita sotto mano: ma naturalmente le cose non filano così lisce, perché due astronavi nemiche vi daranno la caccia e semineranno mine vaganti per cercare di fermarvi.

Ultimo dettaglio: la seconda astronave nemica entrerà in gioco solo dopo che avrete completato la distruzione delle prime otto stelle.

Le istruzioni di gioco e i comandi da usare sono scritti a video all'inizio (righe 2-5 del listato): buon divertimento e occhio alle mine! (L.F.)

Sorting col metodo «Bubble Sort»

L'ordinare dei punti numerici o alfabetici è un compito classico per un computer.

Chiaramente l'ordinamento può essere fatto in ordine crescente od in ordine decrescente.

Per ordinare dei dati esistono molti metodi; in questo numero di Personal Computer vedremo quali sono e quali usare secondo la capacità del computer a nostra disposizione.

Anzitutto ordinare dei dati vuol dire cambiar loro di posizione fisica o logica e per far questo usiamo la tecnica dello scambio. Il procedimento per tale scambio è il seguente:

Confronto Dato (I) con Dato (L) se sono già ordinati proseguo con incrementare I, altrimenti scambio, cioè:

SAVE = DATO (I)
DATO (I) = DATO (L)
DATO (L) = SAVE

Uso cioè una variabile tampone SAVE temporanea.

L'ordinamento Bubble - Sort è il più diffuso ma non è né il più veloce, né quello che occupa meno memorie, ma è il più facile da capire.

Listato - Bubble Sort

```
10 REM BUBBLE SORT
20 INPUT "Numero di dati ";N
30 DIM DATO(N)
40 FOR I=1 TO N
50 INPUT "DATO ";DATO(I)
60 NEXT I
70 REM ORDINAMENTO
80 FOR I=1 TO N-1
90 FOR L=I+1 TO N
100 IF DATO(I)<  =DATO(L) THEN 500
110 REM SCAMBIO
120 SAVE=DATO(I)
130 DATO(I)=DATO(L)
140 DATO(L)=SAVE
500 NEXT L
510 NEXT I
```

Problemi di listato

Cercheremo, in questo articolo, un nuovo modo di listare i programmi per i computers Commodore. Il problema deriva dal fatto che la Commodore usa caratteri speciali, non ASCII, per rappresentare importanti comandi, colori, caratteri e così via. Non è sempre possibile in un listato stampato per differenziare tra i caratteri grafici speciali, uguagliare con stampanti specificamente costruite per questi computer. Per esempio; una linea verticale sottile in un solido blocco nero, può rappresentare alcune funzioni differenti su un Commodore. Piuttosto che cercare di rappresentare la grafica speciale di questi computers, abbiamo stampato un testo che descrive la funzione. Per esempio la funzione HOME sui Commodore 64/VIC 20, che sposta il cursore nell'angolo alto a sinistra dello schermo, è rappresentata graficamente con una S in campo inverso. Proprio quando questa S in campo inverso è «leggibile» dal listato, non rappresenta quello che è accaduto. I nostri listati rimpiazzano la singola S in campo inverso con la stringa (HOME). Il cuore in campo inverso, che rappresenta la funzione pulisci video, può essere listato come (CLEAR).
Una lista completa di queste conversioni di listaggio è riportata di seguito all'articolo per rendere facile le traslazioni. Il listato delle linee che hanno caratteri grafici speciali nel generatore di caratteri EPSON FX 80, sono presentati sotto nel nuovo formato.
Spero che questa nuova tecnica possa rendere i programmi più comprensibili e facili da usare.

Note - Problemi di listato

1) rappresenta il tasto SHIFT.
2) = rappresenta il tasto Commodore nella parte bassa a sinistra della tastiera
3) CTRL rappresenta il tasto CTRL
4) i caratteri grafici sono rappresentati nel listato dai tasti battuti, richiesti per generare il carattere
5) un numero direttamente dopo un (SIMBOLO), indica i multipli del SIMBOLO: (DOWN6) significa battere DOWN per 6 volte.

Problemi di listato

Tabella dei comandi — C/674 - Tastiera

(CLEAR)	^ CLR	;pulitura video
(HOME)	HOME	;ritorno cursore a casa
(INSERT)	^ INST	;inserimento carattere
(DOWN)	CRSR DOWN	;cursore giù
(UP)	^ CRSR UP	;cursore su
(RIGHT)	CRSR RIGHT	;cursore a destra
(LEFT)	^ CRSR LEFT	;cursore a sinistra

COLORI

(BLACK)	CTRL 1 BLK	;nero
(WHITE)	CTRL 2 WHT	;bianco
(RED)	CTRL 3 RED	;rosso
(CYN)	CTRL 4 CYN	;blu verde
(PURPLE)	CTRL 5 PUR	;porpora
(GREEN)	CTRL 6 GRN	;verde
(BLUE)	CTRL 7 BLU	;blu
(YELLOW)	CTRL 8 YEL	;giallo
(RVS)	CTRL 9 RVS ON	;inverso colori
(RVSOFF)	CTRL 0 RVS OFF	;fine inverso colori
(ORANGE)	= 1	;arancio
(BROWN)	= 2	;marrone
(GREY 1)	= 3	;grigio 1
(GREY 1)	= 4	;grigio 1
(GREY 2)	= 5	;grigio 2
(LT GREEN)	= 6	;verde chiaro
(LT BLUE)	= 7	;azzurro
(GREY 3)	= 8	;grigio 3

FUNZIONI = Tasti funzione programmabili

(F1)	f1
(F2)	^ f2
(F3)	f3
(F4)	^f4
(F5)	f5
(F6)	^ f6
(F7)	f7
(F8)	^ f8

CARATTERI SPECIALI

(PI)	π ^ Pi Greco
(POUND)	£ Sterlina
(UP ARROW)	↑ Freccia su
(BACK ARROW)	← Freccia indietro

COMANDI NON PRESENTI SULLA TASTIERA

(DIS-)	CHR$ (8)
(ENB-)	CHR$ (9)
(LOWER CASE)	CHR$ (14)
(UPPER CASE)	CHR$ (142)
(^RETURN)	CHR$ (142)
(DEL)	CHR$ (20)
(SPACE)	CHR$ (160)

NOTE

Crittografare i programmi

Questo programma, realizzato da **Massimo Roberto Garibaldi**, permette la codifica di altri programmi scritti in Basic.
Può essere utile quando si deve lasciare copie di programmi riservati alla portata di estranei.
Il programma gira su un VIC 20, ma può essere facilmente trasportato su qualunque Personal che ne utilizzi il sistema di memorizzazione delle linee Basic.
Tale metodo è basato sulla trasformazione delle singole istruzioni in numeri, chiamati TOKEN, in base ad una tabella memorizzata nel computer e riportata sul manuale dello stesso.
Il programma da me ideato esegue un OR esclusivo di tutti i token del programma, in base ad una chiave alfanumerica che l'operatore dovrà inserire sia durante la codifica che la decodifica.
Il programma inoltre è completamente trasparente all'utente, in quanto si adatta automaticamente alla configurazione di memoria di cui il VIC può essere dotato. Questo è reso possibile dalle linee 1 e 2 del listato, che calcolano rispettivamente la fine del programma e la fine della memoria RAM disponibile.
I valori ottenuti in base ai puntatori del VIC, vengono poi usati alla linea 3 per spostare la fine della memoria e proteggere quindi alcuni valori indispensabili all'esecuzione corretta del programma seguente. Il cuore del programma sta tra la linea 200 e 220. In particolare è la 210 che si occupa della codifica e decodifica vera e propria.
Come già detto essa consiste in un OR esclusivo tra i tokens del programma ed ogni carattere della chiave di codifica. L'OR esclusivo, che non esiste nel Basic del VIC, è stato reso mediante una differenea tra l'OR e l'AND dei valori da manipolare.
Sempre in linea 210, con l'ultima istruzione, viene esclusa la possibilità che tali valori siano eguali, nel qual caso si avrebbe un'errore nella codifica.
È da aggiungere il fatto che il programma assiste l'utilizzatore dall'inizio alla fine. È infatti completamente automatico l'accodamento del programma da modificare al programma principale che, terminata la codifica, si autodistrugge, permettendo di salvare direttamente su nastro o disco il programma cifrato.

L'orologio sveglia con il C64

Prendo spunto dalla lettera di **Massimo Roberto Garibaldi** per pregare cortesemente i nostri lettori di allegare al listato del programma una adeguata spiegazione del suo funzionamento, sia per aiutare i lettori a capirne il meccanismo (copiare pedestremente un programma non aiuta sicuramente ad imparare a programmare) sia per aiutare il sottoscritto a verificarne il funzionamento.
Passiamo ora al programma del nostro lettore.
Il listato tratta di un orologio sveglia che si inserisce in permanenza nell'angolo superiore destro dello schermo e può quindi essere utilizzato durante l'esecuzione o la realizzazione di programmi per tener conto del tempo impiegato, oppure come sveglia vera e propria. Il programma, interamente in linguaggio macchina, basa il suo funzionamento sulla modifica delle routine di interrupt permettendo il continuo aggiornamento basato sul clock esterno.
A questo punto aggiungiamo due righe: può accadere che, anche battendo correttamente il programma, quando si imposta **SYS 832,**

Problemi di listato

```
10 PRINT "(CLEAR) GENERATORE DI CARATTERI"
1010 BY=FX+13*CH:PRINT "(HOME)";
1020 FOR J=7 TO 0 STEP -1:K=2(UP ARROW)J
1440 FOR J=7 TO 0 STEP -1:K=2(UP ARROW)J
2015 PRINT "(HOME,RIGHT8)CARATTERI #";CH;"(LEFT)  (SPACE,HOME)";
2040 IF T$="(POUND)" THEN CP=225:GOTO 2025
2050 IF T$="(UP ARROW)" THEN CP=87:GOTO 2025
3000 PRINT "(HOME)";QQ$;LL$;"(RIGHT)";INPUT CH
3020 PRINT "(HOME)";QQ$;LL$;S9$
4000 PRINT "(CLEAR)";QQ$;"+ ACCESO / - SPENTO / (POUND) MEZZO /
    (UP ARROW) PIENO"
4010 PRINT"(DOWN)F1 LEGGE CARATTERE #"
5130 PD=ASC("."):PRINT"(CLEAR,BLACK)";
5160 CH=0:F1$="(F1)":F2$="(F2)":F3$="(F3)"
5165 F4$="(F4)":F5$= "(F5)":F6$="(F6)":F7$="(F7)"
5170 CL$="(RIGHT)":CR$="(LEFT)":CU$="(UP)"
5175 CD$="(DOWN)":HM$="(HOME)":CS$="(CLEAR)"
5180 ZL$=RT$+"(UP,RIGHT14)"
5190 ZR$="(LEFT17)"
5196 QQ$="(DOWN10)"
5197 LL$="(RIGHT18)"
6010 PRINT"(HOME,DOWN17)";
6210 PRINT"(HOME,DOWN16)";
6300 PRINT"(UP)";EL$;B$;" CARICATO";:GOTO 2010
6600 PRINT RT$;"(UP,RIGHT19)";
6604 PRINT"(LEFT18)";
7200 PRINT QQ$;"(DOWN)";
7210 PRINT RT$;"(RIGHT16)";
7220 PRINT CH;"(LEFT)        (LEFT7)";
```

Listato - Crittografare i programmi

```
          CRITTOGRAFIA
BY      M.R.G. SOFTWARE
          (@) 1983
```

```
0 PRINT"▮▮                              CRITTOGRAFIA                              "
1 I=PEEK(45)+PEEK(46)*256-2:H=INT(I/256):L=I-H*256
2 F=PEEK(55)+PEEK(56)*256-4:HF=INT(F/256):LF=F-HF*256
3 POKE51,LF:POKE55,LF:POKE52,HF:POKE56,HF:POKEF,PEEK(43):POKEF+1,PEEK(44):POKEF+
2,L:POKEF+3,H
4 PRINT"▮▮METTI LA CASSETTA NEL REGISTRATORE POI PREMIUN TASTO":WAIT197,255,64
5 PRINT"▮▮▮▮PΓ43,"L":PΓ44,"H":LΓ":L=PEEK(F):H=PEEK(F+1):PRINT"▮PΓ43,"L":PΓ44,"H"
:GΓ20"
10 POKE631,19:POKE632,17:POKE633,17:POKE634,31:POKE635,13:POKE636,19:POKE637,13:
POKE198,7:END
20 INPUT"▮▮▮▮CHIAVE";K$
25 IFK$=""THEN20
30 F=PEEK(55)+PEEK(56)*256
40 K=PEEK(F+2)+PEEK(F+3)*256+4:C=1
50 GOSUB200:K=K+1:C=C+1
60 IFPEEK(K)=0THENK=K+5:IFPEEK(K-3)=0ANDPEEK(K-4)=0THEN100
80 GOTO50
100 F=PEEK(55)+PEEK(56)*256:L=PEEK(F+2):H=PEEK(F+3):POKE43,L:POKE44,H:END
200 IFLEN(K$)<CTHENC=1
210 P=PEEK(K):P=(PORASC(MID$(K$,C,1)))-(PANDASC(MID$(K$,C,1))):IFP=0THENRETURN
220 POKEK,P:RETURN

READY.
```

non si veda comparire alcun orologio. Niente paura, non è il programma che non funziona, sono solo le cifre dell'orologio che hanno lo stesso colore dello sfondo del teleschermo, e quindi è sufficiente digitare **POKE 982, n** (n = numero del colore desiderato).

Il gioco «Isola» per VIC20 e C64

«Isola» è un grande gioco di strategia statica proposto su Personal Computer n. 1/1983, ed è commercializzato in Italia dalla Ravensburger.

L'obiettivo del gioco è di «isolare» l'avversario muovendo sulla apposita scacchiera come un re di scacchi. Le dimensioni della tavola da gioco sono a scelta dei giocatori e variano da un minimo di 7 x 5 a un massimo di 9 x 7 caselle.

La particolarità cui «Isola» deve il suo nome sta nel fatto che una volta effettuata la mossa, la casella di provenienza viene cancellata dal gioco e resa intransitabile. Il programma richiede i nomi dei giocatori, identificati da simboli diversi, e gestisce le mosse, controllando che le richieste siano lecite: i commenti nei programmi e nelle mappe video spiegano il resto, tuttavia è il caso di ricordare che è possibile muovere solo su caselle ancora transitabili, non è lecito calpestare l'avversario e infine perde chi resta senza mosse valide.

Al termine della partita il calcolatore si complimenta con il vincitore.

Vista anche la dimensione e la semplicità dei listati, nella versione del nostro amico di vecchia data **Andrea Acquaviva** di Rovigo, vi suggeriamo di provare questo gioco, che ha riscosso molti consensi ed è stato considerato uno dei migliori proposti lo scorso anno. Buon divertimento!

(L.F.)

Listato Orologio sveglia

```
110 GOSUB210
120 PRINT"]▓▓** IL TEMPO NON ASPETTA **▓"
130 PRINT"ISTRUZIONI PER L'OROLOGIO:▓"
140 PRINT"SYS 832      :ACCENDE
150 PRINT"SYS 994      :SPEGNE
160 PRINT"POKE 982,N  :CAMBIA COLORE

180 PRINT"F1           :SPEGNE SVEGLIA"
190 GOSUB360:END
200 REM LE ROUTINES POSSONO ESSERE USATE INDIPENDENTI
210 CH=0:FORI=832TO1008
220 READA:POKEI,A:CH=CH+A:NEXT
230 FORI=679TO742:READA:POKEI,A:CH=CH+A:NEXT
240 IFCH-23588THENPRINT"QWHOA...DATA ERROR":STOP:NOTE CHECKSUM
250 INPUT"]▓▓▓▓▓ AM O PM ?";A$:INPUT"▓ ORA ?";H
260 PRINT"▓▓ DIGITA IL MINUTO DI PARTENZA"
270 PRINT"  PREMI RETURN PER INIZIARE▓"
280 IFH>12THENA$="P":H=H-12:GOTO280
290 IFH>9THENH=H+6:REM CONVERSIONE IN BCD
300 IFLEFT$(A$,1)="P"THENH=H+128
310 C=56328:POKEC+3,H:POKEC+1,0
320 INPUTM:M=M+INT(M/10)*6
330 POKEC+2,M:POKEC,0:SYS832:PRINT"▓▓ SE NON VA BENE, PREMI UN TASTO
340 FORI=1TO1000:IFPEEK(198)THENPOKE198,0:SYS994:GOTO250
350 NEXT:RETURN
360 PRINT"QUNDO VUOI LA SVEGLIA?▓"
370 INPUT"AM O PM";A$:A$=LEFT$(A$,1)
380 INPUT"A CHE ORA?";H
390 IFH>12THENA$="P":H=H-12:GOTO390
400 H=H-6*(H>9)-128*(A$="P"):REMCONVERTEIN BCD E METTEINDICATORE AM/PM
410 INPUT"A CHE MINUTO";M
420 M=M+INT(M/10)*6
430 C=56328:POKEC+7,136:POKEC+3,H:POKEC+2,M:POKEC,1:POKEC+7,8:REMSVEGLIA
440 POKE54273,99:POKE54278,240:POKE54276,21
450 POKE54287,2:POKE54290,17:REM SUONO
460 RETURN
470 DATA120,173,20,3,162,89,234,234,234,142,20,3,173,21,3,162,3
480 DATA234,234,234,142,21,3,88,96,173,11,220,170,41,15,24,105,48
490 DATA141,67,4,138,16,4,162,16,16,2,162,1,142,77,4,162,32
500 DATA41,16,240,2,162,49,142,66,4,173,10,220,170,41,15,105,48
510 DATA141,70,4,138,74,74,74,74,24,105,48,141,69,4,173,9,220
520 DATA170,41,15,105,48,141,73,4,138,74,74,74,74,24,105,48,141
530 DATA72,4,173,8,220,105,48,141,75,4,169,32,141,65,4,141,76
540 DATA4,141,79,4,162,14,157,24,4,202,208,250,169,58,141,68,4
550 DATA141,71,4,169,46,141,74,4,169,13,141,78,4,169,1,162,13
560 DATA157,65,216,202,208,250,76,167,2,120,169,49,234,141,20,3,169
570 DATA234,234,141,21,3,88,96
580 DATA173,13,220,41,4,240,3,141,225,2,173,225,2,240,40,165,162
590 DATA106,106,106,41,12,141,32,208,41,4,141,24,212,240,11,162,5
600 DATA189,225,2,157,33,4,202,208,247,165,197,201,4,208,6,142,225
610 DATA2,142,24,212,76,49,234,0,1,12,1,18,13

READY.
```

Listato «Isola» per VIC 20

```
10 REM ******************************
20 REM *                            *
30 REM *         I S O L A          *
35 REM *                            *
40 REM *   PERSONAL COMPUTER CLUB   *
45 REM *                            *
50 REM *      VERSIONE VIC 20       *
70 REM *    DI ANDREA ACQUAVIVA     *
80 REM *                            *
90 REM ******************************
100 REM
110 REM * VARIABILI STRINGA PER I SIMBOLI DELLE DIREZIONI *
120 M$(1)="▣↖▢/ ":M$(2)="▣▣▢█| ":M$(3)="\▣▢▢ ":M$(4)="<- "
130 M$(5)="-> ":M$(6)="▢▣\▢ ":M$(7)="▣|█↑ ":M$(8)="▣/▢ "
140 C$="                      "
150 PRINT"▢";" ▣ ISOLA BY P.C CLUB ■"
155 REM * ISTRUZIONI INIZIALI *
160 PRINT"▣▣▣ PER MUOVERE USARE :"
170 PRINT"▣▣ - 1 = ";M$(1);"  - 2 = ";M$(2):PRINT"▣▣ - 3 = ";M$(3);
180 PRINT"  - 4 = ";M$(4):PRINT"▣▣ - 6 = ";M$(5);"  - 7 = ";M$(6)
190 PRINT"▣▣ - 8 = ";M$(7);"   - 9 = ";M$(8):PRINT:PRINT
195 PRINT" - PREMI UN TASTO -"
200 GETR$:IFR$=""THEN200
210 REM * RICHIESTA NOMI E DIMENS. SCACCHIERA *
220 PRINT"▢";" ▣ ISOLA BY P.C CLUB ■"
230 PRINT"▣▣▣ NOME DEI GIOCATORI":PRINT
240 PRINT:INPUT" 1°    ";P1$
250 PRINT:INPUT" 2°    ";P2$
270 PRINT"▣▣ SCEGLI LE DIMENSIONI  DELLA SCACCHIERA :":PRINT
280 PRINT" ( MIN.7X5, MAX.9X7 )":PRINT
290 INPUT" ORIZZ. ";X
300 PRINT:INPUT" VERT.  ";Y
990 REM * DISEGNO DELLA SCACCHIERA E DEI SIMBOLI *
1000 PRINT"▢";" ▣ ISOLA BY P.C CLUB ■"
1200 A=7748:B=(X*2-2):C=38468
1250 FORR=1TOY:FORI=0TOBSTEP2:POKEA+I,102:POKEC+I,6
1350 NEXT
1400 A=A+44:C=C+44
1450 PRINT:PRINT
1500 NEXT
1550 G1=7680:G2=7834+I
1600 POKEG1,90:POKEG2,83
1640 GOSUB3500:PRINT" CHI GIOCA PER PRIMO ":PRINT
1650 INPUT" '1' O '2' ";P$
1660 PRINT"▢";C$
1670 P=VAL(P$)
1680 IFP<1ORP>2THEN1640
1690 IFP=2THEN1710
1695 REM PRINCIPALE *** 4 LINEE PER LA GESTIONE DEL GIOCO ***
1700 T=G1:GOSUB3500:FORR=1TO5:PRINTC$:NEXT:GOSUB3500:PRINT"MUOVE:( ♦ )";P1$
1705 PRINT:PRINT"MOSSE:":GOSUB2500:POKET,90:G1=T:GOTO1710
1710 T=G2:GOSUB3500:FORR=1TO5:PRINTC$:NEXT:GOSUB3500:PRINT"MUOVE:( ♥ )";P2$
1715 PRINT:PRINT"MOSSE:":GOSUB2500:POKET,83:G2=T:GOTO1700
2490 REM * SUBROUTINE PER INDIVIDUARE LE MOSSE POSSIBILI *
2500 M=0:RESTORE
```

Listato «Isola» per VIC20

```
2510 FORJ=1TO8
2520 READD
2530 IFPEEK(T+D)=102THENPRINTM$(J);:M=1
2540 NEXT
2680 IFM=0THENGOSUB3500:GOTO3550
2690 REM * SUBROUTINE PER CONSENTIRE LA MOSSA EFFETTUATA *
2700 GETA$:IFA$=""THEN2700
2750 IFA$="5"THEN2900
2760 IFVAL(A$)<1ORVAL(A$)>9THEN2900
2780 FORJ=1TO9
2785 READD
2790 IFVAL(A$)=JANDPEEK(T+D)=102THENPOKET,32:T=T+D:RETURN
2800 NEXT
2810 REM * MESSAGGIO D' ERRORE *
2820 RESTORE:FORJ=1TO8:READD:NEXT
2900 GOSUB3500
3000 FORR=1TO10:PRINT"▯MOSSA NON VALIDA":FORR1=1TO200:NEXT:PRINT"▯";C$:NEXT
3010 GOTO2700
3500 PRINT"▩":FORR=1TO16:PRINT:NEXT
3510 RETURN
3520 REM * SCRIVE IL VINCITORE *
3550 IFPEEK(T)=83THENV$=P1$
3560 IFPEEK(T)=90THENV$=P2$
3600 FORR=1TO3:PRINTC$:NEXT
3610 GOSUB3500
3620 FORR=1TO20:PRINTC$:PRINT" ▯▩ V I N C E   ";V$;" ▦"
3624 FORR1=1TO100:NEXT:PRINT"▯";:NEXT
3630 PRINT"▩▩ GIOCATE ANCORA (S/N)"
3640 GETN$:IFN$=""THEN3640
3650 IFN$="N"THENPRINT"▩     ARRIVEDERCI !"
3660 IFN$="S"THENPRINT"▯":GOTO270
3700 DATA42,44,46,-2,2,-46,-44,-42,42,44,46,-2,0,2,-46,-44,-42
```

Listato «Isola» per C64

```
10 REM ********************************
20 REM *                              *
30 REM *         I S O L A            *
35 REM *                              *
40 REM *    PERSONAL COMPUTER CLUB    *
45 REM *                              *
50 REM *        COMMODORE 64          *
60 REM *                              *
70 REM ********************************
80 REM
100 REM *** PROGRAMMA ISOLA DI ANDREA ACQUAVIVA ***
110 REM * VARIABILI STRINGA PER I SIMBOLI DELLE DIREZIONI *
120 M$(1)=" ▩▯/ ":M$(2)=" ▩▯▮|| ":M$(3)=" \▩▯ ":M$(4)=" <- "
130 M$(5)=" -> ":M$(6)=" ▩\▯ ":M$(7)=" ▩▯▮↑ ":M$(8)=" ▩/▯ "
140 C$="                                  "
145 POKE53281,0
150 PRINT"▯";" ▩ I S O L A  BY PERSONAL COMPUTER CLUB ▦"
155 REM * ISTRUZIONI INIZIALI *
160 PRINT"▩▩▩ PER MUOVERE USARE I TASTI INTORNO AL 5"
170 PRINT"▩▩  - 1 = ";M$(1);"     - 2 = ";M$(2):PRINT"▩▩  - 3 = ";M$(3);
180 PRINT"     - 4 = ";M$(4):PRINT"▩▩  - 6 = ";M$(5);"     - 7 = ";M$(6)
```

Listato Isola per C64

```
190 PRINT"▒▒ - 8 = ";M$(7);"    - 9 = ";M$(8):PRINT:PRINT
195 PRINT:PRINT"  - PREMI UN TASTO PER CONTINUARE -"
200 GETR$:IFR$=""THEN200
210 REM * RICHIESTA NOMI E DIMENS. SCACCHIERA *
220 PRINT"▒";" ▒ I S O L A  BY PERSONAL COMPUTER CLUB ▒"
230 PRINTTAB(10)"▒▒▒NOME DEI GIOCATORI":PRINT
240 PRINT:INPUT" 1° GIOCATORE ";P1$
250 PRINT:INPUT" 2° GIOCATORE ";P2$
270 PRINT"▒▒SCEGLI LE DIMENSIONI DELLA SCACCHIERA : "
280 PRINT"( MINIMO 7 X 5 , MASSIMO 9 X 7 )":PRINT
290 INPUT"ORIZZ. ";X
300 PRINT:INPUT"VERT.  ";Y
990 REM * DISEGNO DELLA SCACCHIERA E DEI SIMBOLI *
1000 PRINT"▒";" ▒ I S O L A  BY PERSONAL COMPUTER CLUB ▒"
1200 A=1154:B=(X*2-2):C=55426
1250 FORR=1TOY:FORI=0TOBSTEP2:POKEA+I,102:POKEC+I,1
1350 NEXT
1400 A=A+80:C=C+80
1450 PRINT:PRINT
1500 NEXT
1550 G1=1394:G2=1312+I
1600 POKEG1,90:POKEG2,83
1640 GOSUB3500:PRINT"CHI GIOCA PER PRIMO ";
1650 INPUT" '1' O '2' ";P$
1660 PRINT"▒";C$
1670 P=VAL(P$)
1680 IFP<1ORP>2THEN1640
1690 IFP=2THEN1710
1695 REM PRINCIPALE *** 4 LINEE PER LA GESTIONE DEL GIOCO ***
1700 T=G1:GOSUB3500:PRINTC$;C$:PRINT"▒MUOVE:    ( ♦ )    ";P1$;C$
1705 GOSUB3500:PRINT"MOSSE : ";:GOSUB2500:POKET,90:G1=T:GOTO1710
1710 T=G2:GOSUB3500:PRINTC$;C$:PRINT"▒MUOVE:    ( ♥ )    ";P2$;C$
1715 GOSUB3500:PRINT"MOSSE : ";:GOSUB2500:POKET,83:G2=T:GOTO1700
2490 REM * SUBROUTINE PER INDIVIDUARE LE MOSSE POSSIBILI *
2500 M=0:RESTORE
2510 FORJ=1TO8
2520 READD
2530 IFPEEK(T+D)=102THENPRINTM$(J);:M=1
2540 NEXT
2680 IFM=0THENGOSUB3500:GOTO3550
2690 REM * SUBROUTINE PER CONSENTIRE LA MOSSA EFFETTUATA *
2700 GETA$:IFA$=""THEN2700
2750 IFA$="5"THEN2900
2760 IFVAL(A$)<1ORVAL(A$)>9THEN2900
2780 FORJ=1TO9
2785 READD
2790 IFVAL(A$)=JANDPEEK(T+D)=102THENPOKET,32:T=T+D:RETURN
2800 NEXT
2810 REM * MESSAGGIO D' ERRORE *
2820 RESTORE:FORJ=1TO8:READD:NEXT
2900 GOSUB3500
3000 FORR=1TO10:PRINT"▒MOSSA NON VALIDA":FORR1=1TO200:NEXT:PRINT"▒";C$:NEXT
3010 GOTO2700
3500 PRINT"▒":FORR=1TO18:PRINT:NEXT
3510 RETURN
3520 REM * SCRIVE IL VINCITORE *
```

Listato Isola per C64

```
3550 IFPEEK(T)=83THENV$=P1$
3560 IFPEEK(T)=90THENV$=P2$
3600 FORR=1TO3:PRINTC$:NEXT
3610 PRINT"▒" FORR=1TO17:PRINT:NEXT
3620 FORR=1TO20:PRINTC$:PRINTTAB(5)"▒▒   V I N C E    ";V$;"   ▒"
3624 FORR1=1TO100:NEXT:PRINT"▒";:NEXT
3630 PRINT"▒▒GIOCATE  ANCORA ( S / N )"
3640 GETN$:IFN$=""THEN3640
3650 IFN$="N"THENPRINT"▒ARRIVEDERCI !"
3660 IFN$="S"THENPRINT"▒":GOTO270
3700 DATA78,80,82,-2,2,-82,-80,-78,78,80,82,-2,0,2,-82,-80,-78
```

Listato Prugne

```
READY.

 10 PRINT"          -------------------"
 20 PRINT"          ***     PRUNE     ***"
 30 PRINT"          -------------------"
 35 PRINT
 40 PRINT"            ANTONIO MARTINO "
 50 PRINT"          MAURIZIO MASCHERONI "
 60 PRINT
 65 PRINT"                MILANO"
 70 PRINT"             NOVEMBRE 1983"
 80 PRINT
 90 PRINT
 100 GOTO50000
 200 PA=1064:SC=0:V=54272:C4=0:NS=0
 210 C(1)=10:FORX=2TO10:C(X)=2:NEXTX:NF=3
 220 W1=9:W2=5:W3=11:W4=11:W5=6:W6=3:FORX=1TO10:S(X)=0:NEXT
 300 POKE53281,W6:GOSUB42000:GOSUB43000
 1000 NS=NS+1
 1100 REM
 1102 REM POSIZIONO PRUGNE
 1104 REM
 1110 PF=1563:C1=0:C2=0:NG=0:FX=0:FY=-1:NT=0:FORX=1TO3:SG(X)=1:NEXT
 1120 P(1)=1923:P(2)=1363:P(3)=1554:P(4)=1572:P(5)=1314:P(6)=1332
 1140 P(7)=1158:P(8)=1168:P(9)=1758:P(10)=1768
 1292 REM
 1294 REM ESPLODO VIDEO ED ENTRO IN BATTAGLIA
 1296 REM
 1300 GOSUB40100
 1350 GOSUB30000
 1372 REM
 1374 REM USCITA DALLA BATTAGLIA
 1375 REM C9=0 NULLA DI DECISIVO - TORNO A BATTAGLIARE
 1376 REM C9=1 DISTRUTTO UN FUNGO
 1377 REM C9=2 DISTRUTTE TUTTE LE PRUGNE - SCHERMO SUCCESSIVO
 1378 REM C9=3 DISTRUTTI TUTTI FUNGHI - FINE PARTITA
 1379 REM
 1380 IFC9=0THEN1350
 1382 IFC9=2THENPRINTCHR$(31):PRINT"▒▒▒▒▒▒▒▒▒▒▒          OTTIMO "N$"!"
 1383 IFC9=2THENFORT=1TO1500:NEXT
 1385 PRINT"▒▒":PRINTCHR$(31);
 1390 ONC9GOTO2100,1500,2000
 1492 REM
 1494 REM TUTTE PRUGNE K.O.
 1496 REM
 1500 EX=INT((RND(1)*100)+1)+50*NS:SC=SC+EX
 1510 IFSC>=PEANDC4=0THENC4=1:NF=NF+1
 1520 PRINT"▒▒▒▒▒▒                ▒SITUAZIONE▒"
 1530 PRINT"▒              PUNTI   "SC-EX
 1540 PRINT"▒              BONUS   "EX
 1550 PRINT"▒              TOTALE ▒"SC"▒"
 1560 PRINT"▒▒▒▒▒               ('C' PER CONTINUARE)"
 1570 GOSUB44600
 1580 GETB$:IFB$=""THEN1580
 1590 IFB$<>"C"THEN1580
 1600 NP=NW:FORX=1TONW:S(X)=0:NEXT:GOSUB43000:GOTO1000
 1992 REM
 1994 REM TUTTI FUNGHI K.O.
 1996 REM
 2000 PRINT"▒▒▒▒▒                ▒ GAME  OVER ▒"
 2010 PRINT"▒▒             ▒COMPLIMENTI!▒"
 2020 PRINT"▒▒             PUNTI ▒"SC"▒"
 2030 PRINT"▒▒▒▒▒              ('C' PER CONTINUARE)"
 2050 GETC$:IFC$=""THEN2050
```

Le «prugne» per il C.64

Vi proponiamo «Prune» un interessante programma realizzato da **Antonio Martino.** L'originalità del gioco ed una grafica, rendono questo programma molto interessante. Lasciamo la parola al nostro lettore per descrizione e commenti.

Considerazioni generali

«Prune» è stato ideato per un Commodore 64, ed è scritto interamente in Basic. È un gioco a tutto schermo. Ogni entità mobile presente nel gioco può sconfinare dal video e rientrare dalla parte opposta. Non si tratta di colpire certi bersagli sparando proiettili evitando la collisione diretta, ma si deve cercare, tranne alcune eccezioni, proprio la collisione.

Lo scopo del gioco è di manovrare un **funghetto** marrone al fine di catturare delle prugne che corrono sullo schermo.

I comandi a disposizione del giocatore sono quelli per la direzione del proprio fungo, che si muove in 8 direzioni, secondo il presente schema:

```
   U
 Y   I
 G • K
 B   M
   N
```

A disposizione anche i tasti «H» e «J», indifferentemente, per sganciare le **trappole** (cuoricini verdi).

Le prugne

Il numero delle prugne che appare all'inizio di ogni schermo è scelto dal giocatore all'inizio della partita ed è compreso tra 1 e 10. Tutte le prugne sono rosse, tranne una, la **superprugna,** che è rosa. Se si sceglie di giocare contro una sola prugna questa sarà, per diritto divino, una superprugna. La superprugna è più pericolosa, più difficile da catturare, ma ovviamente vale anche più punti se catturata.

Per catturare le prugne bisogna semplicemente cozzare contro di esse con il fungo o sperare che cadano in un cuoricino verde.

Le prugne si muovono in 8 direzioni diverse. La direzione di ognuna di esse, indipendente dalle altre, è scelta a caso.

La superprugna cambia sempre direzione, mentre le altre prugne mantengono una direzione costante per un certo numero di movimenti (la costanza diminuisce con l'aumentare delle schermate).

Circa 1 volta su 6 ogni prugna può lasciare una **trappola** (due occhioni blu), che naturalmente il fungo deve evitare.

Ogni tanto, la prugna lancia un **«prugnino»** grigio. Questo avviene circa 1 volta su 16 per le prugne normali e circa 1 volta su 8 per la superprugna. In ogni caso, il numero dei prugnini presenti contemporaneamente sullo schermo non può essere superiore a 3.

Se la prugna incontra una propria trappola (gli occhioni blu) viene distrutta (e questo fa comodo al fungo).

Se la prugna, bontà sua, incontra un'altra prugna o un prugnino, si blocca e blocca pure tutta la fase d'attacco della propria «squadra» (prugne, trappole, prugnini); la mossa passa al fungo che ha in pratica qualche movimento extra per scorrazzare indisturbato e fare terra bruciata intorno a sé.

Listato Le Prugne

```
2055 IFC$<>"C"THEN2050
2060 GOTO3000
2065 POKE53272,21
2070 POKE53281,6:PRINT"█":CLR:END
2092 REM
2094 REM FUNGO DISTRUTTO
2096 REM
2100 PRINT"██████                  █SITUAZIONE█"
2120 PRINT"███                PUNTI:  █"SC"█"
2130 PRINT"███                FUNGHI:  "NF
2140 PRINT"█████                 ('C' PER CONTINUARE)"
2160 GETB$:IFB$=""THEN2160
2165 IFB$<>"C"THEN2160
2170 GOSUB43000:GOTO1110
2992 REM
2994 REM TABELLA RECORD
2996 REM
3000 FORX=1TO10:IFSC>=R(X)THEN3140
3020 NEXTX
3030 GOTO3300
3140 IFX=10GOTO3180
3150 FORJ=10TOX+1STEP-1
3160 R(J)=R(J-1):N$(J)=N$(J-1):P$(J)=P$(J-1):R$(J)=R$(J-1)
3170 NEXTJ
3180 N$(X)=N$:R(X)=SC:P$(X)=""
3190 FORK=1TONW:P$(X)=P$(X)+CHR$(113):NEXTK
3210 X$="":X$=STR$(SC):M4=0:R$(X)=""
3220 R$(X)=STR$(SC)
3300 PRINTCHR$(31):PRINT"███
3310 PRINT"██ PRUNE        THE TOP TEN        ":PRINT"█"
3320 FORK=1TO10:IFR(K)=0THEN3360
3325 PRINT"█";
3330 PRINT" █"K"█  "N$(K);
3340 PRINTTAB(18);R$(K);:PRINTTAB(29)"█"P$(K)"█"
3350 NEXTK
3360 PRINT"█████████████████████████  UN'ALTRA PARTITA?";
3370 GETC$:IFC$=""THEN3370
3380 IFC$<>"S"ANDC$<>"N"THEN3370
3390 IFC$="S"THEN200
3400 GOTO2065
29992 REM ------------------------------
29994 REM            BATTAGLIA
29996 REM ------------------------------
29997 REM
29998 REM FASE MOVIMENTO PRUGNE
29999 REM
30000 C9=0:FORK=1TONW:C7=0
30010 IFS(K)=1THEN30600
30020 IFK>1ANDNC(K)=PPTHENNC(K)=0
30030 NC(K)=NC(K)+1
30040 IFNC(K)=1ORK=1THENGOSUB40200
30050 NU=P(K)+DX(K)+40*DY(K)
30060 IFNU>2023THENNU=NU+1064-2024
30080 IFNU<1064THENNU=NU+2024-1064
30090 NV=P(K)
30100 IFPEEK(NU)=81THEN30800
30110 IFPEEK(NU)=124THEN30800
30120 POKENV,32:POKENV+V,W6
30150 IFPEEK(NU)=90THENGOSUB41300:C7=1:M1=34:M2=175:GOSUB44200
30153 IFC9=2THEN31100
30155 IFC7=1THEN30800
30160 IFPEEK(NU)=83THENPOKENU,81:POKENU+V,C(K):GOSUB40300:C7=1
30161 IFC7=1THENM1=34:M2=175:GOSUB44200
30180 IFC9=2THEN31100
30190 IFC7=1THEN30800
30192 P(K)=NU
30200 POKEP(K),81:POKEP(K)+V,C(K)
30250 IFINT(RND(1)*6)+1>1THEN30300
30270 POKENV,87:POKENV+V,W5
30300 IFNG=3THEN30600
30320 IFK=1ANDINT(RND(1)*8)+1=1THEN30380
30340 IFK>1ANDINT(RND(1)*16)+1=1THEN30380
30360 GOTO30600
30380 FORW=1TO3:IFSG(W)=1THEN30400
30390 NEXTW
30400 GX(W)=DX(K)*(-1):GY(W)=DY(K)*(-1):
30410 Q(W)=0:NG=NG+1:G(W)=NV:SG(W)=0
30420 PG(W)=INT(RND(1)*5+1)*NS+2
30430 POKEG(W),124:POKEG(W)+V,W3
30600 NEXTK
30602 REM
30604 REM MOVIMENTO PRUGNINI
30606 REM
30610 FORK=1TO3:C7=0
30620 IFSG(K)=1THEN30740
30630 Q(K)=Q(K)+1
30640 IFQ(K)>PG(K)THENM1=22:M2=227:GOSUB44200:GOSUB41000
30645 IFC5=1THEN31040
30647 IFC7=1THEN30740
30650 HG=G(K)
```

Listato Le Prugne

```
30660 POKEHG,32:POKEHG+V,W6
30670 G(K)=G(K)+GX(K)+40*GY(K)
30680 IFG(K)>2023THENG(K)=G(K)+1064-2024
30690 IFG(K)<1064THENG(K)=G(K)+2024-1064
30700 IFPEEK(G(K))=83THENM1=34:M2=175:GOSUB44200:GOSUB40700:GOTO30740
30710 IFPEEK(G(K))=90THENPOKEG(K),124:POKEG(K)+V,W3:GOTO31040
30720 IFPEEK(G(K))=32THENPOKEG(K),124:POKEG(K)+V,W3:GOTO30740
30730 GOTO30670
30740 NEXTK
30792 REM
30794 REM MOVIMENTO FUNGO
30796 REM
30800 D1=1
30805 GETB$
30810 IFB$="G"THENFX=-1:FY= 0:GOTO30890
30820 IFB$="Y"THENFX=-1:FY=-1:GOTO30890
30830 IFB$="U"THENFX= 0:FY=-1:GOTO30890
30840 IFB$="I"THENFX= 1:FY=-1:GOTO30890
30850 IFB$="K"THENFX= 1:FY= 0:GOTO30890
30860 IFB$="M"THENFX= 1:FY= 1:GOTO30890
30870 IFB$="N"THENFX= 0:FY= 1:GOTO30890
30880 IFB$="B"THENFX=-1:FY= 1:GOTO30890
30890 FF=PF:FU=PF+FX+40*FY
30900 IFFU>2023THENFU=FU+1064-2024
30910 IFFU<1064THENFU=FU+2024-1064
30912 POKEFF,32:POKEFF+V,W6
30914 IFB$<>"H"ANDB$<>"J"THEN30920
30915 IFNT=2*NWTHEN30920
30916 POKEFF,83:POKEFF+V,W2:NT=NT+1
30920 IFPEEK(FU)=83THENNT=NT-1:GOTO30970
30930 IFPEEK(FU)=87THENPOKEFU,90:POKEFU+V,W1:GOTO31040
30940 IFPEEK(FU)=124THENPOKEFU,90:POKEFU+V,W1:GOTO31040
30950 IFPEEK(FU)=81THENPOKEFU,90:POKEFU+V,W1:GOSUB41400:M1=34:M2=175:GOSUB44200
30960 IFC9=2THEN31100
30970 PF=FU:POKEPF,90:POKEPF+V,W1
30980 D1=D1+1:IFD1<=2THEN30805
31020 GOTO31100
31040 M1=17:M2=37:GOSUB44200:C9=1:NF=NF-1:IFNF=0THENC9=3
31050 PRINTCHR$(31):PRINT"[illegible]            PECCATO "N$"!":FORT=1TO1500:NEXT
31100 RETURN
40092 REM
40094 REM ESPLODO VIDEO DI PARTENZA
40096 REM
40100 GOSUB40500
40110 FORX=1TONW:IFS(X)=1THEN40140
40130 POKEP(X),81:POKEP(X)+V,C(X)
40140 NEXTX
40150 POKEPF,90:POKEPF+V,W1:FORX=1TONW:NC(X)=0:NEXT
40180 PP=11-NS:IFPP<3THENPP=3:RETURN
40192 REM
40194 REM ESTRAZIONE CASUALE MOVIMENTO PRUGNE
40196 REM
40200 A=INT(RND(1)*8)+1
40210 IFA=1THENDX(K)=-1:DY(K)= 0:RETURN
40220 IFA=2THENDX(K)=-1:DY(K)=-1:RETURN
40230 IFA=3THENDX(K)= 0:DY(K)=-1:RETURN
40240 IFA=4THENDX(K)= 1:DY(K)=-1:RETURN
40250 IFA=5THENDX(K)= 1:DY(K)= 0:RETURN
40260 IFA=5THENDX(K)= 1:DY(K)= 1:RETURN
40270 IFA=7THENDX(K)= 0:DY(K)= 1:RETURN
40280 IFA=8THENDX(K)=-1:DY(K)= 1:RETURN
40285 RETURN
40292 REM
40294 REM PRUGNA CADUTA IN TRAPPOLA
40296 REM
40300 POKENU,32:POKENU+V,W6:NT=NT-1
40310 IFK>1THENSC=SC+INT(100/NP)*NS
40320 IFK=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*3)+2)
40330 IFSC>=PEANDC4=0THENC4=1:NF=NF+1
40340 NP=NP-1:S(K)=1:NC(K)=0:GOSUB40500
40360 IFNP=0THENC9=2
40370 RETURN
40492 REM
40494 REM TESTATA CON PUNTI
40496 REM
40500 PRINTCHR$(31):PRINT"[illegible]
40520 PRINT"[illegible] PRUNE  SCHERMO "NS"   PUNTI: "SC:RETURN
40692 REM
40694 REM PRUGNINO CADUTO IN TRAPPOLA
40696 REM
40700 POKEG(K),32:POKEG(K)+V,W6
40710 NT=NT-1:NG=NG-1:SG(K)=1:Q(K)=0:PG(K)=0
40720 SC=SC+INT((RND(1)*50)+10)*NS:GOSUB40500:RETURN
40992 REM
40994 REM ESPLOSIONE PRUGNINO
40996 REM
41000 C5=0:C7=1
41020 FORI=1TO3
41040 FORH=G(K)-1+((I-2)*40)TOG(K)-1+((I-2)*40)+2
41060 IFH=G(K)THEN41110
```

Se la prugna incappa in un cuoricino verde, viene distrutta. Il punteggio assegnato aumenta col diminuire delle prugne presenti e con l'aumentare degli schermi; per l'intrappolamento della superprugna il punteggio così calcolato viene casualmente raddoppiato, triplicato, o quadruplicato.

Se la prugna incappa nel fungo, viene catturata: punteggio calcolato nel modo già visto, ma aumentato perché abbiamo saputo dirigere il fungo nella direzione giusta.

Quando tutte le prugne vengono catturate viene assegnato un «bonus» che aumenta con l'aumentare degli schermi, e si passa allo schermo successivo.

I prugnini

I prugnini grigi sono perfidi in sommo grado e non possono essere distrutti dal fungo. Bisogna evitarli e sperare che cadano in un cuoricino verde, unico modo per catturarli.

Al momento del lancio da parte della prugna, il prugnino corre in direzione opposta a quella della propria prugna per un numero di movimenti casuale, che aumenta con l'aumentare degli schermi.

Se incontra un fungo, lo distrugge.

Se incontra un altro prugnino, una prugna, o gli occhioni blu lasciati dalle prugne, il prugnino non si scompone più di tanto ed esegue un balzo al di là dell'ostacolo, proseguendo, come se niente fosse, nella sua marcia alla ricerca del nostro povero funghetto.

Arrivato alla fine della sua corsa (quando cioè ha esaurito il numero dei movimenti assegnatogli casualmente), il prugnino si autodistrugge, esplodendo intorno a sè della **«spazzatura»** grigia; questa spazzatura distrugge tutto (anche i nostri cuoricini verdi), tranne, naturalmente,

entità amiche sue (prugne, ecc.)

L'unico modo per debellare questo essere furibondo è sperare che cada in un cuoricino verde prima di esplodere. Il punteggio che viene assegnato è casuale (come tutti i punteggi) ma comunque abbastanza alto e aumenta con l'aumentare degli schermi.

I funghi

Abbiamo a disposizione 3 funghi. Al conseguimento di un certo punteggio (che varia a seconda del numero delle prugne contro le quali stiamo giocando) ci viene assegnato un funghetto extra. Alla cattura dell'ultimo fungo a nostra disposizione, il gioco finisce.

Il numero massimo di cuoricini verdi che possiamo lasciare è pari al doppio del numero delle prugne contro le quali stiamo giocando. Se il fungo passa su un proprio cuoricino verde, lo distrugge senza altre conseguenze. Se il numero di cuoricini verdi diminuisce (perché ci siamo passati sopra col fungo o perché hanno fatto il loro dovere catturando prugne e prugnini), allora possiamo depositarne altri, senza però poter superare, ovviamente, il massimo consentito.

Onde facilitare l'azione del funghetto, per ogni movimento di prugne e prugnini, vengono eseguiti due movimenti del fungo.

Fasi della battaglia

Ogni fase logica della battaglia si compone, nell'ordine, dei seguenti movimenti.

1 - Movimento della superprugna
2 - Movimento delle altre prugne
3 - Movimento dei prugnini
4 - Movimento del fungo (ripetuto 2 volte)

Naturalmente, per ogni movimento vengono controllate tutte le possibili collisioni onde intraprendere le azioni

Listato Le Prugne

```
41062 Z=H
41064 IFZ>2023THENZ=Z+1064-2024
41066 IFZ<1064THENZ=Z+2024-1064
41070 IFPEEK(Z)=87ORPEEK(Z)=124ORPEEK(Z)=81THEN41110
41080 IFPEEK(Z)=83THENNT=NT-1
41090 IFPEEK(Z)=90THENC5=1
41100 POKEZ,102:POKEZ+V,W4
41105 IFC5=1THEN41200
41110 NEXTH:NEXTI
41130 POKEG(K),32:POKEG(K)+V,W6:NG=NG-1:SG(K)=1:Q(K)=0:PG(K)=0
41140 FORI=1TO3
41150 FORH=G(K)-1+((I-2)*40)TOG(K)-1+((I-2)*40)+2
41152 Z=H
41154 IFZ>2023THENZ=Z+1064-2024
41156 IFZ<1064THENZ=Z+2024-1064
41160 IFPEEK(Z)<>102THEN41180
41170 POKEZ,32:POKEZ+V,W6
41180 NEXTH:NEXTI
41200 RETURN
41292 REM
41294 REM COLLISIONE PRUGNA/FUNGO
41296 REM
41300 IFK>1THENSC=SC+INT(100/NP)*NS+INT(100/NP)
41330 IFK=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*4)+2)+INT(100/NP)
41340 NP=NP-1:S(K)=1:NC(K)=0:IFNP=0THENC9=2
41360 IFSC>=PEANDC4=0THENC4=1:NF=NF+1
41370 GOSUB40500:RETURN
41392 REM
41394 REM COLLISIONE FUNGO/PRUGNA
41396 REM
41400 FORT=1TONW:IFS(T)=1THEN41420
41410 IFFU=P(T)THEN41440
41420 NEXTT
41440 IFT>1THENSC=SC+INT(100/NP)*NS+INT(100/NP)
41450 IFT=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*4)+2)+INT(100/NP)
41460 NP=NP-1:S(T)=1:NC(T)=0
41470 IFNP=0THENC9=2
41480 IFSC>=PEANDC4=0THENC4=1:NF=NF+1
41490 GOSUB40500:RETURN
41992 REM
41994 REM ISTRUZIONI PRINCIPALI
41996 REM
42000 GOSUB43200:PRINTCHR$(31)
42010 PRINT"                      PRUNE ":PRINT
42030 PRINT"SCOPO DEL GIOCO E' DI MANOVRARE UN"
42040 PRINT"FUNGO PER COLPIRE DELLE PRUGNE."
42050 PRINT"TRA ESSE C'E' UNA SUPERPRUGNA, PIU'"
42060 PRINT"CATTIVA MA DI MAGGIOR VALORE."
42070 PRINT"DEVI EVITARE I 'PRUGNINI' SPARATI DALLE"
42080 PRINT"PRUGNE. CON I TASTI 'H' O 'J' PUOI"
42090 PRINT"LASCIARE DELLE TRAPPOLE, E DEVI EVITARE"
42100 PRINT"DI CADERE IN QUELLE LASCIATE DALLE "
42110 PRINT"PRUGNE. FUNGO EXTRA A "PE" PUNTI."
42120 PRINT"PUOI MUOVERTI IN 8 DIREZIONI,"
42130 PRINT"SECONDO IL SEGUENTE SCHEMA:"
42140 PRINT
42150 PRINT"                              U"
42160 PRINT"          GOOD                 Y I"
42170 PRINT"                             G ♦ K      "
42180 PRINT"          LUCK                  B M "
42190 PRINT"                              N"
42210 PRINT"(UN TASTO PER INIZIARE)";
42220 GETC$:IFC$=""THEN42220
42450 RETURN
42992 REM
42994 REM COUNTDOWN
42996 REM
43000 PZ=1355:PW=1636
43020 PRINTCHR$(31)
43030 PRINT"                    COUNTDOWN"
43040 PRINT"               CREDIT: "
43060 FORM=1TONP
43080 POKEPZ-2+2*M,81:POKEPZ-2+2*M+V,2
43090 NEXTM
43100 FORM=1TONF
43110 POKEPW-2+2*M,90:POKEPW-2+2*M+V,W1
43120 NEXTM
43130 FORZ=1346TOPZ+2*NP:D2=0
43135 IFPEEK(Z)=81THEND2=1
43140 POKEZ-1,32:POKEZ-1+V,W6
43150 POKEZ,90:POKEZ+V,W1:IFD2=1THENGOSUB44000
43155 FORT=1TO100:NEXTT
43160 NEXTZ:PRINT""
43170 RETURN
43192 REM
43194 REM SCELTA NUMERO PRUGNE
43196 REM
43200 PRINTCHR$(31)
43210 PRINT"                    PRUNE ":PRINT""
43230 INPUT"CONTRO QUANTE PRUGNE VUOI GIOCARE";F$
```

Listato Le Prugne

```
43240 NW=INT(VAL(F$)):IFNW<1ORNW>10THEN43210
43250 NP=NW:PE=3000+500*(11-NP)
43251 PRINT"■■■■■■■■■■■■"
43252 PRINT"■■■■■■■■■■■■":INPUT"COME TI CHIAMI";N$
43256 IFLEN(N$)>11THEN43251
43260 RETURN
43292 REM
43294 REM NOTA PER COUNTDOWN
43296 REM
44000 S=54272
44010 FORL=0TO24:POKES+L,0:NEXT
44020 POKES+1,130
44030 POKES+5,9
44040 POKES+15,30
44050 POKES+24,15
44060 POKES+4,21
44070 FORT=1TO100:NEXT:POKES+4,20
44090 FORT=1TO100:NEXT:POKES+24,0
44091 RETURN
44092 REM
44094 REM NOTA PER COLLISIONE
44096 REM
44200 POKE54296,15:MA=0
44205 POKE54277,9:POKE54276,17
44206 FORT=1TO300:NEXT
44207 IFMA=1GOTO44220
44210 POKE54273,M1:POKE54272,M2:POKE54276,0
44215 MA=1:GOTO44205
44220 POKE54276,0:POKE54273,0:POKE54272,0
44230 RETURN
44592 REM
44594 REM ROUTINE MUSICALE PER BONUS
44596 REM
44600 FORL=1TO12
44620 POKE54296,15:POKE54276,65:POKE54277,190:POKE54275,15:POKE54274,15
44640 POKE54273,M%(L,1):POKE54272,M%(L,2)
44650 FORX=M%(L,3)-50TOM%(L,3)-20:POKE54278,136:NEXT
44660 FORT=1TOM%(L,3):NEXT
44670 POKE54273,0:POKE54272,0:POKE54276,0
```

conseguenti (esempio: blocco delle fasi 2 e 3 se una prugna incontra un'altra prugna, ecc.)
Good Luck! (W.P.)

Inserire il titolo sul C64

«Titolo» va inserito all'inizio di ogni programma o utilizzato come subroutine.
Dato il run, compare il titolo del programma, già opportunamente inserito alla riga 13 come A$.
Su sfondo nero, il titolo è scritto in blu con tre cornici: bianca, rossa e verde. A questo punto non ci resta che premere un tasto e partirà il nostro programma normalmente.

Lista delle variabili listato «Inserire titolo»

A$ - titolo da visualizzare e, alla fine, argomento del get.
T - posizione di partenza della cornice.
K - contatore della lunghezza della cornice.

Listato Inserire il titolo

```
RUN12
1 :
2 :          T I T O L O
3 :          -----------
4 :
5 :          PER  CBM 64
6 :
7 :        ANTONIO PONTORIERI
8 :     VIA PORTA CALCINARA, 41
9 :    27100 PAVIA - TEL.0382/37173
10 :
11 :
12 POKE53280,0:POKE53281,0:PRINT"■■";CHR$(142);
13 A$="***************":REM *==> QUI VA INSERITOIL TITOLODELPROGRAMMA.<
14 PRINT"■■■■■■■■■■";:T=16-LEN(A$)/2
15 PRINTTAB(T+1)"■__";:FORK=1TOLEN(A$):PRINT"_";:NEXT:PRINT"__"
16 PRINTTAB(T)" ■r-";:FORK=1TOLEN(A$):PRINT"-";:NEXT:PRINT"-¬■ "
17 PRINTTAB(T)"■ ■|■¯";:FORK=1TOLEN(A$):PRINT"¯";:NEXT:PRINT"¯■|■ "
18 PRINTTAB(T)" ■|■ ■"A$"■ ■|■ "
19 PRINTTAB(T)"■ ■|■_";:FORK=1TOLEN(A$):PRINT"_";:NEXT:PRINT"_■|■ "
20 PRINTTAB(T)" ■L-";:FORK=1TOLEN(A$):PRINT"-";:NEXT:PRINT"-┘■ "
21 PRINTTAB(T+1)"■¯¯";:FORK=1TOLEN(A$):PRINT"¯";:NEXT:PRINT"¯¯■"
22 GETA$:IFA$=""THEN22
23 REM *==> DA QUI PUO' COMINCIARE IL TUO PROGRAMMA <==*

READY.
```

Spiegazione delle istruzioni listato «Inserire titolo»

12. Pulisce lo schermo e imposta lo sfondo nero. Il chr$ (142) imposta il set di caratteri standard (maiuscolo-grafico).
13. Al posto degli asterischi va inserito il titolo, che può avere una lunghezza qualsiasi (purché non sia più lungo della riga video)
14. Posiziona il cursore al centro del video, posizione che dipende dalla lunghezza del titolo.
15. Forma la parte superiore della cornice esterna di colore verde.
16. Forma la parte superiore della cornice centrale di colore bianco.
Attenzione! Prima della 'E' che imposta il bianco (CTRL+2), si ha il carattere che si ottiene con COMMODORE+M, così anche dopo la freccia in su (CTRL+6) si ha il simbolo ottenuto con CMDR+G.
17. Tra virgolette abbiamo: CTRL+6, CMDR+M, CTRL+2, SHIFT+B, CMEDR+3, SHIFT+0; Poi: SHIFT+P, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.
18. CMDR+M, CTRL+2, SHIFT+B, CMDR+3, CMDR+G, CMDR+7; Poi: CMDR+3, CMDR+M, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.
19. CTRL+6, CMDR+M, CTRL+2, SHIFT+B, CMDR+3, SHIFT+L; Poi: SHIFT+chiocciolina, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.
20. CMDR+M, CTRL+2, ecc..., CTRL+6, CMDR+G.
21. Chiude la cornice più esterna.
22. Attende che venga premuto un tasto.
23. Da qui puoi inserire il tuo programma, che avrà ora un aspetto molto più 'professional', non trovi?
Naturalmente, chi vorrà avere il programma già registrato su cassetta, non dovrà fare altro che telefonarmi o scrivermi.

Incorniciare lo schermo

Presentiamo nel **listato 1** una piccola utility per VIC 20, adattabile con poche modifiche al C 64, per stampare una cornice attorno allo schermo. È stata realizzata dal coordinatore del club di Personal Computer di Bra: Massimo Fumero.

Nel **listato 2**, Massimo mostra invece come stampare una frase in una certa posizione dello schermo, individuata da X e Y:

Il programma fa uso della funzione LEFT$ (stringa, N), che permette di stampare i primi N caratteri della stringa partendo da sinistra.

In questo caso, la stringa è composta da 22 caratteri corrispondenti a «cursor down».

LEFT$ («22 cusor down, 13) stampa i primi 13 caratteri della stringa partendo da sinistra, quindi 13 «cursor down»; il cursore verrà quindi posizionato sulla 13esima riga.
Questo piccolo trucchetto vi permette di aggirare il problema della mancanza di un'istruzione VTAB.
Se il trucco dovesse essere usato più volte all'interno di un programma, consiglio di definire una stringa = «22 cursor down» all'inizio, di usare una variabile per la tabulazione verticale e una per quella orizzontale, e di richiamare il trucchetto come una subroutine.

☆☆☆☆☆☆☆☆☆☆☆☆☆☆

Auguri

☆☆☆☆☆☆☆☆☆☆☆☆☆☆

Incorniciare lo schermo

Listato 1

```
10 GOSUB 500
250 END
500 REM *** CORNICE SCHERMO ***
505 POKE 36879,13 : PRINT "(CTRL-2)".
510 I = 7680 : J = 484 : S = 81 : PRINT "(SHIFT-CLR)"
515 FOR K = I TO I + 21 : POKE K,S : NEXT
520 FOR K = I + 22 TO I + 22*21 STEP 22
525 POKE K,S : POKE K + 21,S : NEXT
530 FOR K = 1 + J TO I + J + 21 : POKE K,S : NEXT
535 PRINT "(HOME)"
540 RETURN
```

Listato 2

```
600 REM *** CURSORE ***
605 X = 13 : Y = 3
610 PRINT "(HOME)" LEFT$ ("(22 cursor down)" , Y) TAB (X);
620 PRINT "PC - CLUB"
```

Commenti al listato 1

Linea 505:	pone lo schermo nero, il bordo verde e il cursore bianco
Linea 510:	variabili per adattare la subroutine al C 64 I = inizio locazioni di schermo J = numero di caratteri dall'angolo alto sinistro a quello basso sinistro dello schermo S = codice del carattere con cui verrà stampata la cornice, in questo caso il pallino (codice dec. 81)
Linea 515:	stampa il margine superiore della cornice
Linee 520/525:	stampa i bordi laterali della cornice
Linea 530:	stampa il bordo inferiore della cornice

Commenti al listato 2

Linea 605:	variabili di tabulazione. In questo caso la scritta PC - CLUB verrà stampata sulla 13esima colonna (X) e sulla 3a riga (Y)
Linea 610:	porta il cursore all'angolo superiore destro e scende alla 3a riga

Un archivio «fonetico»

In molte applicazioni dei computer è necessario archiviare liste di nomi. Ad esempio clienti e fornitori per il commercio, passeggeri per le agenzie di viaggio, clienti per i professionisti e pazienti per gli ospedali. Questo è un programma per un direttorio telefonico ed illustra alcune delle tecniche utilizzabili in archivi di ricerca ed amministrativi.

Il programma è scritto per il Commodore Pet a 40 colonne, ma non utilizza caratteristiche Basic di macchine specifiche se non nel formattatore di schermo, descritto nel testo. Il programma può funzionare in 8K ma con un direttorio limitato.

Uno degli inconvenienti incontrati nella ricerca in un archivio di nomi è quello di non conoscere con precisione il nome cercato. Un esempio, il nome «Clarke», può essere capito come «Clark», «Clerk» o «Clerke». È chiaro che quando si cerca in un indice è utile conoscere tutti i nomi che sono foneticamente simili.

Il codice Soundex è stato progettato per questo scopo: è compito del programma codificare un nome nel suo Soundex equivalente, nel seguente modo:

1 – la prima lettera del nome è anche la prima del codice

2 – le lettere seguenti sono rimpiazzate in questo modo:
B,F,P e V - rimpiazzate con P
C,G,J,K,Q,S,X e Z - rimpiazzate con S
M e N - rimpiazzate con M
L e R - codificate senza modifiche
A,E,I,O,U,W,H e Y - non codificate
tutte le consonanti con pronuncia similare sono raggruppate

3 - una sequenza ininterotta di lettere con lo stesso valore di codifica viene rimpiazzata

Figura 1

WATT	GURNEY
WITHE	GRIM
WAITE	GOREN
WHITE	GREEN
WYATT	
TUTTI CODIFICATI CON WTAA	TUTTI CODIFICATI CON GRMA

con un singolo codice, sebbene non sia inclusa la prima lettera del nome. Può essere inclusa cambiando la linea di lettura 2000
SD$=LEFT$(NM$,1):L$=SD$:N=O

4 - il codice è forzatamente lungo quattro caratteri, e viene troncato aggiungendo alla destra la lettera A.

La procedura per codificare è nelle linee da 2000 a 2160. Nell'esempio di **figura 1** i nomi foneticamente simili sono codificati in modo identico, però le regole della pronuncia inglese insegnano che questa non è la regola. «Belvoir» viene codificato BLPR, però può essere pronunciato «Bearer», che ha il codice BPRA. Così «Bough» è codificato BSAA

Figura 2

	Subscript	Key	Pointer
	1	JKL	0
	2	unused	5
	3	DEF	7
FF →	4	unused	2
	5	unused	0
FU →	6	ABC	3
	7	GHI	1

——— used list
– – – free list

ma viene anche pronunciato «Boff» o «Bow», che hanno il codice BPAA e BAAA.

Il codice Soundex è valido per cercare nomi in un piccolo direttorio telefonico personale, ma può solo formare una parte di un algoritmo di ricerca per un archivio più vasto. Generalmente le liste sono cercate per stadi successivi, con criteri di ricerca molto restrittivi e adatti ad ogni stadio. Così, il primo stadio può essere esaminato per un esatto riscontro sul cognome, iniziali, sesso e data di nascita. Se il riscontro richiesto è errato, il secondo stadio cerca sul Soundex sesso e anno di nascita. Il metodo di ricerca dipende dall'importanza assegnata alla ricerca esatta e alla richiesta di riscontri.

Varie strutture di dati possono essere usate per creare un indice. La più semplice è una lista seriale o file. L'ultimo record della lista può essere seguito da nuove aggiunte, per cui la lista viene dichiarata grande a sufficienza per ulteriori espansioni. Le cancellazioni richiedono l'indice per cercare l'ingresso da togliere, e, queste, collocano nel record un valore speciale come Cancellato o «», la stringa nulla. Se vi sono molte cancellazioni, bisogna aggiungere una procedura per recuperare i record cancellati. L'indice deve esaminare ogni record per tutti i riscontri richiesti.

Una struttura molto pratica è la lista anulare. È usata nei programmi ed è largamente impiegata nelle strutture di dati comprensivi e nei sistemi di base dati commerciali. Ad ogni record è aggiunto un puntatore che indica la posizione del prossimo record in sequenza, il puntatore corrisponde al valore del numero del record appropriato. Dato che in Basic non è possibile mischiare variabili stringa e numeriche, bisogna inserire i puntatori in una lista separata, PT%, che è parallela alla linea dei dati principali, TPS.

All'inizio tutti gli elementi della lista sono assegnati a una lista libera. Il puntatore del primo elemento della lista è contenuto in FF e ogni elemento PT% è messo a puntare il seguente, in questo modo:
PT%(1)=2
PT%(2)=3
L'ultimo elemento PT% contiene zero, indicando il fine lista. Un altro puntatore, FU, dà il numero del primo record usato. La struttu-

Le variabili del listato «Archivio fonetico»

NN – numero di record assegnati all'indice
TP$(NN) – indice telefonico
PT%(NN) anelli record
RC$ – ingresso indice
NM$ – nome
SX$ – sesso
AD$ – indirizzo
TL$ – numero telefonico
SD$ – codice Soundex
NP – puntatore per il successivo record
IP – puntatore del record precedente
CP – puntatore del record corrente
FU – puntatore del primo record dell'indice
FF – puntatore del primo record inutilizzato
IP$ – chiave per azionare Input
LF$ – nome del file da caricare
SF$ – nome del file da salvare
UD – bandiera di aggiornamento, fissata su 1 se il file è creato o emendato
ER – bandiera di errore
ND – numero di record da visualizzare
SE$ – carattere separatore, “ ”
CD$ – stringa movimento cursore
A$, N, I, N$, L$, C$, J, K, L – variabili di uso comune.

Archivio fonetico - Procedure

– **Caratteri** di movimento cursore, particolari del Pet, sono visualizzati con codici interni a parentesi quadre. I codici usati sono: CD, cursore giù, HOM, cursore a casa e CLS, pulire schermo.

– **Procedura Master,** linee 200-310. Chiama l'inizio e le procedure di istruzione. Accetta l'ingresso dei tasti d'azione e chiama la procedura corrispondente.

– **Inizio,** linee 400-410. Fissa le costanti e pone UD=O, mostrando che all'inizio l'indice non richiede salvataggi. Open 1,0 in linea 410 apre la tastiera e un ingresso periferico. I dati possono essere accettati dalla tastiera con Input 1. Premendo il tasto Return si ha il vantaggio che Ready sul video è ignorato. Con Input 1 si ha lo svantaggio di non poter usare il promemoria e il Return che completa l'ingresso non produce una nuova linea sullo schermo. Questi svantaggi vengono facilmente eliminati e sono un piccolo prezzo da pagare in un programma che non visualizza Ready.

– **Istruzioni,** linee 600-665. Direttamente visualizzate sullo schermo.

– **Procedura di carico,** linee 800-890. Inizialmente NN è zero; se non lo è, un'indice è già stato creato/caricato. Il file nominato è aperto e letto e l'utente se lo desidera può estenderlo. Il record aggiunto viene inanellato nei record già esistenti dalle linee 860-870.

– **Procedura di controllo aggiunte,** linee 1000-1170. La linea 1000 assicura che non possono essere aggiunti record prima che il file sia stato caricato o creato. I dettagli del record da aggiungere sono nominati nelle linee 1010-1100, che controllano gli errori. Non si possono immettere stringhe con le virgole, in quanto nel Basic una virgola è un separatore. La subroutine 2000, richiamata dalla linea 1110, crea il codice Soundex. La linea 1120 crea il record concatenando le parti separate. La subroutine 2200, richiamata dalla linea 1130, aggiunge all'indice un nuovo record. Le linee 1440-1170 raggirano la condizione di File Completo trovata dalla subroutine 2200. L'utente è invitato a salvare il file e a riavviare il programma ed espandere il file quando viene caricato.

– **Procedura di ricerca,** linee 1200-1290. Le linee 1200-1210 accettano la chiave di ricerca. Le linee 1220-1240 scorrono l'indice un record alla volta. La subroutine 2400 restituisce CP con il puntatore del successivo record sequenziale. La linea 1250 se non ci sono riscontri, visualizza un messaggio che riferisce il risultato negativo della ricerca. La linea 1270 stampa il record riscontrato e controlla che lo schermo non sia completo. Se lo è, le linee 1280-1290 permettono all'utente di esaminare i riscontri prima di visualizzare altri riscontri. Specifico del Pet è il Wait 59410,4,4 che aspetta sia premuta la barra spaziatrice. Se questa istruzione non è battuta correttamente il Pet sospende l'istruzione Wait bloccandone il ciclo.

Alcuni codici alternativi sono:

1290 GET A$:IF A$ <> « » THEN 1290
1300 PRINT «[CLS]»:GOTO 1240

– **Salvataggio indice,** linee 1400-1430. L'indice viene salvato su un file, su cassetta di registrazione. Sul registratore, la virgola separatrice tra le variabili viene forzata includendola nelle istruzioni PRINT 2.

– **Nuovo indice,** linee 1600-1650. Sono richiesti i dettagli del nuovo file e viene fissata la lista corrispondente. Le linee 1640-1650 inanellano tutti gli elementi della lista nella lista inutilizzata.

– **Procedura di fine,** linee 1800-1850. Il file tastiera viene chiuso. Se l'indice è stato cambiato all'utente viene data l'opportunità di salvarlo.

– **Soundex,** linee 2000-2160. Vedere la descrizione nel testo.

– **Ingresso Nuove Aggiunte,** linee 2200-2350. Linea 2200, se FF=0 non vi sono record inutilizzati e il nuovo record non può essere aggiunto. La linea 2210 posiziona il record in TP$(FF) e fissa un ciclo per trovare la locazione corretta in cui aggiungere il record; le linee 2220-2240 danno corpo al ciclo. Il ciclo quando si è trovato il punto corretto o quando tutti i record utilizzati sono stati esaminati e NP=0, li lascia entrambi. La linea 2250 aggiunge all'inizio dell'indice. Le linee 2260-2270 aggiungono all'indice il nuovo record. La linea 2280 aggiunge a un'indice vuoto. Le linee 2290-2300 aggiungono alla fine dell'indice.

– **Prossimo record,** linea 2400. Ripristina IP, CP e NP.

– **Visualizzazione dei record,** linee 2600-2690. La parte principale della procedura, linee 2610-2670, suddivide RC$ in nome indirizzo e numero telefonico. L'attuale visualizzazione è alla linea 2680. La linea 2690 incrementa il contatore dei record visualizzati.

– **Cancellazione,** linee 2800-2910. La chiave del record da cancellare è immessa nella linea 2800, e l'indice analizzato per la ricerca nelle linee 2810-2840. Le linee 2850-2880 danno l'opportunità all'utente di rinunciare alla cancellazione. Le linee 2890-2910 cancellano il record come descritto nel testo.

– **Chiave d'ingresso,** linee 3000-3090. Il nome e il sesso sono accettati dalle linee 3020-3070, mentre il codice Soundex è creato dalla linea 3080.

ra è mostrata in figura 2.

Ogni nuovo record è assegnato alla posizione data da FF, il primo record della lista dei record inutilizzati; è il numero 7 nell'esempio. Il puntatore della lista è quindi portato al valore del puntatore corrispondente a questo record, nell'esempio 18.

Il nuovo record è incastrato nella lista dei record usati di modo che i record sono mantenuti in sequenza.

Quindi il puntatore del record della chiave GHK è fissato sul numero del nuovo record 7 e il puntatore di questo record è messo a puntare il record 4, precedentemente puntato da GHK.

Cancellare il record è più semplice. Il puntatore della lista libera FF è fissato sul numero del record da cancellare, mentre il puntatore di questo record è fissato sul valore precedente di FF, ponendo così il record cancellato alla testa della lista libera. Il record che prima puntava il record cancellato è puntato sul record precedentemente puntato dal record cancellato. Entrambe queste procedure richiedono codifiche addizionali per trattare con aggiunte e cancellazioni da e per l'inizio e la fine della lista. La ricerca è eseguita sequenzialmente seguendo i puntatori. Questo assicura che le chiavi siano analizzate in una sequenza ascendente e la ricerca è completata quando la chiave del record oltrepassa la chiave cercata.

La struttura a lista anulare è idonea ai record contenuti in RAM ma richiede l'aggiunta di uno o più indici quando si utilizza con il backing-store.

Può formare la base di un tale sistema, ed ha la virtù di permettere aggiunte e cancellazioni senza ristrutturare il file.

Il programma per eseguire queste procedure è stato preparato in maniera modulare, semplificando così la codifica, la prova e le successi-

ve espansioni. L'arte della codifica sta nel risolvere il problema e stabilirne la soluzione. Il primo passo è quello di definire il problema: la codifica può essere lasciata più tardi nella programmazione del ciclo. Se il programma è stato ben pianificato fin dal principio, codificare diventa, per la maggior parte, un processo meccanico. Il passo successivo è quello di ridurre la soluzione a un fissaggio di moduli. A questo punto potete scegliere di usare uno pseudo linguaggio strutturato o una flowchart. La principale richiesta è che i moduli devono essere ben definiti, funzionali e sufficientemente piccoli, per essere chiari. È possibile codificare ogni uno in un ragionevole numero di linee.

Nel Basic, ogni modello si aspetta che il modulo controllore possa essere codificato in una subroutine. Siccome è chiaro lo scopo dei moduli, e non possono essere troppo lunghi, non presentano difficoltà di codifica. Inoltre possono essere codificati in TOP-DOWN. Ogni subroutine è inizialmente codificata con una matrice, che può essere un semplice Return una linea per stampare il nome seguita da Return. Poi ogni subroutine è codificata rimpiazzando la matrice, così il programma è gradualmente costruito. La matrice è necessaria dove può fissare valori in variabili per simulare la loro funzione, le procedure sono aggiunte e provate una per volta. Così il processo continua e le procedure più comuni possono essere usate frequentemente, cosicché possiate diventare più fiduciosi sulla loro correttezza.

Listato Archivio fonetico

```
Suoni similari
--------------

200 GOSUB 400
210 GOSUB 600
220 GET IP$:IF IP$="I" THEN 210
230 IF IP$="L" THEN GOSUB 800:GOTO 300
240 IF IP$="A" THEN GOSUB 1000:GOTO 300
250 IF IP$="F" THEN GOSUB 1200:GOTO 300
260 IF IP$="S" THEN GOSUB 1400:GOTO 300
270 IF IP$="N" THEN GOSUB 1600:GOTO 300
280 IF IP$="D" THEN GOSUB 2800:GOTO 300
290 IF IP$="E" THEN GOSUB 1800:END
295 GOTO 220
300 PRINT "  IMMETTERE IL CODICE PER LA SUCCESSIVA FUNZIONE,"
310 PRINT "OPPURE 'I' PER LE ISTRUZIONI":GOTO 220
400 SE$="•":CD$="                              ":UD=0
410 OPEN 1,0:RETURN
600 PRINT " LE FUNZIONI DISPONIBILI SONO:-"
605 PRINT "  L - CARICARE UN'INDICE TELEFONICO"
610 PRINT "S - SALVARE UN'INDICE TELEFONICO"
615 PRINT "N - INSERIRE UN NUOVO INDICE TELEFONICO"
620 PRINT "A - AGGIUNGERE UN'INGRESSO ALL'INDICE"
625 PRINT "F - CERCARE L'INDICE"
630 PRINT "D - CANCELLARE UN'ENTRATA DELL'INDICE"
635 PRINT "E - FINE DEL PROGRAMMA"
```

Listato Archivio fonetico

```
640 PRINT "I - TORNA A QUESTO VIDEO"
645 PRINT "  ORA, O IN RISPOSTA AL PROMEMORIA:-"
650 PRINT "  IMMETTERE IL CODICE PER LA SUCCESSIVA FUNZIONE,"
655 PRINT "OPPURE 'I' PER LE ISTRUZIONI"
660 PRINT " IMMETTERE UNA DELLE LETTERE CITATE"
665 RETURN
800 IF NN<>0 THEN PRINT " UN INDICE E' GIA' STATO CARICATO":RETURN
805 PRINT " IMMETTERE IL NOME FILE":INPUT#1,LF$:PRINT
810 OPEN 2,1,0:INPUT#2,FU,FF,NN
820 PRINT "  CI SONO";NN;"RECORD NEL FILE"
830 PRINT " QUANTI IN PIU'? ";:INPUT#1,A$:N=VAL(A$):PRINT
840 DIM TP$(NN+N),PT%(NN+N)
850 IF N=0 THEN 880
860 FOR I=NN+1 TO NN+N-1:PT%(I)=I+1:NEXT
870 PT%(NN+N)=FF:FF=NN+1
880 FOR I=1 TO NN:INPUT#2,PT%(I),TP$(I):NEXT
890 NN=NN+N:CLOSE 2:RETURN
1000 IF NN=0 THEN PRINT " USARE 'N' O 'L' PER CREARE O CARICARE
     UN FILE":RETURN
1010 PRINT " INSERIRE NOME, COGNOME. NON USARE":PRINT "VIRGOLE":PRINT
1020 INPUT#1,NM$:PRINT
1030 PRINT " INSERIRE SESSO('M' O 'F' O 'O' PER":PRINT "COMMERCIO
     ECC.)";
1040 INPUT#1,SX$:PRINT
1050 IF SX$="M" OR SX$="F" OR SX$="O" THEN 1070
1060 PRINT "'M' O 'F' O 'O' PER FAVORE. RIPETERE":GOTO 1040
1070 PRINT " INSERIRE INDIRIZZO. NON USARE VIRGOLE"
1080 PRINT:INPUT#1,AD$:PRINT
1090 PRINT " INSERIRE NUMERO TELEFONICO"
1100 PRINT:INPUT#1,TL$:PRINT
1110 GOSUB 2000
1120 RC$=SD$+NM$+SE$+AD$+SE$+TL$
```

Listato Archivio fonetico

```
1130 GOSUB 2200
1140 IF ER=0 THEN UD=1:RETURN
1150 PRINT " FILE PIENO. INSERIRE 'S' PER SALVARLO, QUINDI"
1155 PRINT " 'E' PER FINIRE. RIAVVIARE IL PROGRAMMA. USARE"
1160 PRINT " 'L' PER CARICARE IL FILE ED ESPANDERLO"
1165 PRINT " QUANDO AVETE CHIESTO DI VOLERE PIU'"
1170 PRINT " RECORD":RETURN
1200 GOSUB 3000:ND=0
1210 IF ER<>0 THEN RETURN
1220 GOSUB 2400
1230 IF SD$=LEFT$(TP$(CP),5) THEN 1270
1240 IF NP<>0 THEN 1220
1250 IF ND=0 THEN PRINT " NESSUN RISCONTRO":RETURN
1260 PRINT " TUTTI I RISCONTRI TROVATI":RETURN
1270 GOSUB 2600:IF ND-6*INT(ND/6)<>0 THEN 1240
1280 PRINT "  BATTERE SPAZIO PER CONTINUARE"
1290 WAIT 59410,4,4:PRINT " ":GOTO 1240
1400 PRINT " INSERIRE IL NOME FILE":INPUT#1,SF$:PRINT
1410 OPEN 2,1,1:PRINT#2,FU;",";FF;",";NN
1420 FOR I=1 TO NN:PRINT#2,PT%(I);",";TP$(I):NEXT
1430 UD=0:CLOSE 2:RETURN
1600 IF NN<>0 THEN PRINT " AVETE GIA' UN FILE":RETURN
1604 PRINT " QUANTI RECORDS RICHIEDETE? ";
1605 INPUT#1,N$:NN=VAL(N$):PRINT
1610 IF NN<20 OR NN>250 THEN PRINT " 20-250 E' IL LIMITE.
     REINSERIRE";:GOTO 1605
1620 DIM TP$(NN),PT%(NN)
1630 FU=0:FF=1
1640 FOR I=1 TO NN-1:PT%(I)=I+1:NEXT
1650 PT%(NN)=0:RETURN
```

Listato Archivio fonetico

```
1800 IF UD<>1  THEN 1850
1810 PRINT "  AVETE CAMBIATO IL FILE. VOLETE"
1820 PRINT "SALVARLO?"
1830 GET A$:IF A$="N" THEN 1850
1840 IF A$<>"Y" THEN 1830
1845 GOSUB 1400
1850 CLOSE 1:RETURN
2000 SD$=LEF$(NM$,1):L$=" " :N=0
2010 FOR I=2 TO LEN(NM$):A$=MID$(NM$,I,1)
2020 IF A$="B" OR A$="F" OR A$="P" OR A$="V" THEN C$="P":GOTO 2090
2030 IF A$="C" OR A$="G" OR A$="J" OR A$="K" THEN C$="S":GOTO 2090
2035 IF A$="T" OR A$="D" THEN C$="T":GOTO 2090
2040 IF A$="Q" OR A$="S" OR A$="X" OR A$="Z" THEN C$="S":GOTO 2090
2050 IF A$="M" OR A$="N" THEN C$="M":GOTO 2090
2060 IF A$="L" OR A$="R" THEN C$=A$:GOTO 2090
2070 IF A$=" " THEN I=99:GOTO 2110
2080 L$=A$:GOTO 2110
2090 IF C$=L$ THEN 2110
2100 SD$=SD$+C$:N=N+1:L$=C$
2110 NEXT
2120 IF N=3 THEN 2150
2130 IF N<3 THEN SD$=SD$+LEFT$("AAA",3-N):GOTO 2150
2140 SD$=LEFT$(SD$,4)
2150 SD$=SD$+SX$
2160 RETURN
2200 ER=0:IF FF=0 THEN R=1:RETURN
2210 NP=FU:IP=0:CP=0:TP$(FF)=RC$
2220 IF NP=0 THEN 2280
2230 GOSUB 2400:A$=LEFT$(TP$(CP),5)
2240 IF SD$>A$ THEN 2220
2250 IF IP=0 THEN FU=FF:GOTO 2270
```

Listato Archivio fonetico

```
2260 PT%(IP)=FF
2270 N=PT%(FF):PT%(FF)=CP:FF=N:GOTO 2350
2280 IF CP=0 THEN FU=FF:GOTO 2300
2290 PT%(CP)=FF
2300 N=PT%(FF):PT%(FF)=NP:FF=N:GOTO 2350
2350 RETURN
2400 IP=CP:CP=NP:NP=PT%(CP):RETURN
2600 RC$=TP$(CP):L=LEN(RC$):J=0:K=0
2610 FOR I=6 TO L
2620 IF MID$(RC$,I,1)<>"•" THEN 2650
2630 IF J=0 THEN J=I:GOTO 2650
2640 K=I
2650 NEXT
2660 NM$=MID$(RC$,6,J-6):AD$=MID$(RC$,J+1,K-J-1)
2670 TL$=RIGHT$(RC$,L-K)
2680 PRINT " ";NM$;TAB(20);TL$:PRINT AD$
2690 ND=ND+1:RETURN
2800 GOSUB 3000:IF ER<>0 THEN RETURN
2810 GOSUB 2400
2820 IF SD$=LEFT$(TP$(CP),5) THEN 2850
2830 IF NP<>0 AND SD$>=LEFT$(TP$(CP),5) THEN 2810
2840 PRINT " PIU' NESSUN RISCONTRO":RETURN
2850 PRINT "      ":GOSUB 2600
2860 PRINT "  CANCELLATE QUESTO 'RECORD? ('S' O'N')"
2870 GET A$:IF A$="N" THEN 2830
2880 IF A$<>"S" THEN 2870
2890 IF IP=0 THEN FU=PT%(CP):GOTO 2910
2900 PT%(IP)=PT%(CP)
2910 PT%(CP)=FF:FF=CP:TP$(CP)=" ": UD=1:RETURN
3000 ER=0:IF NN=0 THEN PRINT " USATE 'N' O 'L' PER CREARE/CARICARE
     UN FILE":ER=1:RETURN
```

Listato Archivio fonetico

```
3010 IF FU=0 THEN PRINT " NESSUN RECORD SUL FILE":ER=1:RETURN
3020 PRINT " INSERITE NOME E COGNOME. NON USATE":PRINT "VIRGOLE":
     PRINT
3030 INPUT#1,NM$:PRINT
3040 PRINT " INSERIRE SESSO ('M' O 'F' O 'O' PER":PRINT "COMMERCIO
     ECC.)";
3050 INPUT#1,SX$:PRINT
3060 IF SX$="M" OR SX$="F" OR SX$="O" THEN 3080
3070 PRINT " 'M' O 'F' O 'O' PER FAVORE. RIPETERE":GOTO 3050
3080 GOSUB 2000:IP=0:CP=0:NP=FU
3090 PRINT " ":RETURN
```

I caratteri suonano sul VIC 20

Questa piccola routine proposta da **Bruno Montresor**, di Verona gira sul Commodore VIC 20 e permette la visualizzazione dei caratteri accompagnati da un suono. È un effetto insolito che può essere utile in molti casi (per esempio nella presentazione di un gioco).

Il contenuto della variabile A$ a riga 30-40 può essere cambiato a piacere e deve contenere il messaggio da visualizzare. ■

☆☆☆☆☆☆☆☆☆☆☆☆☆

☆ **Auguri** ☆

☆☆☆☆☆☆☆☆☆☆☆☆☆

Listato I caratteri suonano

```
1 REM ** CARATTERI SONORI **
2 REM **   PER  VIC  20   **
10 POKE36879,8:PRINTCHR$(5)
20 PRINT"■"
29 REM ** INIZIO ROUTINE **
30 A$="QUESTA ROUTINE STAMPA SU SCHERMO CARATTERI E SUONI.  "
40 A$=A$+"SI POSSONO AGGIUNGERE MESSAGGI A VOLONTA'.  BUON LAVORO! "
100 FORQ=1TOLEN(A$):S$=MID$(A$,Q,1):PRINTS$;
110 POKE36875,200:POKE36878,15
120 FORI=1TO15:PRINT"■ ■";:NEXTI
130 POKE36875,0:POKE36878,0
140 NEXTQ
141 REM ** FINE ROUTINE **

READY.
```

OLTRE LE BARRIERE DELL'HARD E DEL SOFT.

SVI™
SPECTRAVIDEO

il computer del grande standard MSX



Distributore per l'Italia **COMTRAD** Divisione Computers Tel. (0566) 424348 TLX 623481 COMTRD I